J 4

```
   AAAAAAAAA     NNN         NNN     AAAAAAAAA     LLL              YYY         YYY  ZZZZZZZZZZZZZZZ
   AAAAAAAAA     NNN         NNN     AAAAAAAAA     LLL              YYY         YYY  ZZZZZZZZZZZZZZZ
   AAAAAAAAA     NNN         NNN     AAAAAAAAA     LLL              YYY         YYY  ZZZZZZZZZZZZZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL              YYY         YYY              ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL              YYY         YYY              ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL              YYY         YYY              ZZZ
AAA         AAA  NNNNNN      NNN  AAA         AAA  LLL                 YYY   YYY              ZZZ
AAA         AAA  NNNNNN      NNN  AAA         AAA  LLL                 YYY   YYY              ZZZ
AAA         AAA  NNNNNN      NNN  AAA         AAA  LLL                 YYY   YYY              ZZZ
AAA         AAA  NNN   NNN   NNN  AAA         AAA  LLL                    YYY              ZZZ
AAA         AAA  NNN   NNN   NNN  AAA         AAA  LLL                    YYY              ZZZ
AAA         AAA  NNN   NNN   NNN  AAA         AAA  LLL                    YYY              ZZZ
AAAAAAAAAAAAAAA  NNN      NNNNNN  AAAAAAAAAAAAAAA  LLL                    YYY           ZZZ
AAAAAAAAAAAAAAA  NNN      NNNNNN  AAAAAAAAAAAAAAA  LLL                    YYY           ZZZ
AAAAAAAAAAAAAAA  NNN      NNNNNN  AAAAAAAAAAAAAAA  LLL                    YYY           ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL                    YYY        ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL                    YYY        ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL                    YYY        ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLLLLLLLLLLLLLL        YYY        ZZZZZZZZZZZZZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLLLLLLLLLLLLLL        YYY        ZZZZZZZZZZZZZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLLLLLLLLLLLLLL        YYY        ZZZZZZZZZZZZZZZ
```

D 13

```
**FILE**ID**RMS3IDX
```

```
RRRRRRRR    MM      MM    SSSSSSSS    333333      IIIIII    DDDDDDDD    XX      XX
RRRRRRRR    MM      MM    SSSSSSSS    333333      IIIIII    DDDDDDDD    XX      XX
RR      RR  MMMM  MMMM  SS          33      33      II      DD      DD  XX      XX
RR      RR  MMMM  MMMM  SS          33      33      II      DD      DD  XX      XX
RR      RR  MM  MM  MM  SS                  33      II      DD      DD    XX  XX
RR      RR  MM  MM  MM  SS                  33      II      DD      DD    XX  XX
RRRRRRRR    MM      MM    SSSSSS          33        II      DD      DD      XX
RRRRRRRR    MM      MM    SSSSSS          33        II      DD      DD      XX
RR  RR      MM      MM          SS          33      II      DD      DD    XX  XX
RR  RR      MM      MM          SS          33      II      DD      DD    XX  XX
RR    RR    MM      MM          SS  33      33      II      DD      DD  XX      XX      ....
RR    RR    MM      MM          SS  33      33      II      DD      DD  XX      XX      ....
RR      RR  MM      MM  SSSSSSSS      333333      IIIIII    DDDDDDDD    XX      XX      ....
RR      RR  MM      MM  SSSSSSSS      333333      IIIIII    DDDDDDDD    XX      XX      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0 %title 'RMS3IDX - Analyze Things for Prolog 3 Indexed Files'
0         module rms3idx  (
1                         ident='V04-000') = begin
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! Facility:     VAX/VMS Analyze Facility, Analyze Things for Prolog 3
1 !
1 ! Abstract:     This module is responsible for analyzing various structures
1 !               in prolog 3 indexed files.  Those routines that are common
1 !               to prolog 2 and 3 can be found in RMS2IDX.
1 !
1 !
1 ! Environment:
1 !
1 ! Author: Paul C. Anagnostopoulos, Creation Date: 26 June 1981
1 !
1 ! Modified By:
1 !
1 !       V03-007 PCA1011         Paul C. Anagnostopoulos  1-Apr-1983
1 !               Change the message prefix to ANLRMS$_ to ensure that
1 !               message symbols are unique across all ANALYZEs.  This
1 !               is necessitated by the new merged message files.
1 !
1 !       V03-006 PCA1007         Paul C. Anagnostopoulos 10 Feb 1983
1 !               Add support for recovery unit items in the primary data
1 !               and SIDR records.  This required a new routine to calculate
1 !               the lengths of the various parts of a primary data record,
1 !               since that calculation has become diabolically complex.
1 !
1 !       V03-006 PCA1001         Paul C. Anagnostopoulos 11-Oct-1982
1 !               Add support for prologue 3 SIDRs.
1 !
```

```
;   58          0058  1 !         V03-005 PCA0100          Paul C. Anagnostopoulos  1-Oct-1982
;   59          0059  1 !                 Remove code that displayed the last duplicate bucket
;   60          0060  1 !                 pointer in the bucket trailer.  That pointer was
;   61          0061  1 !                 not used in V3, but the code was left in.
;   62          0062  1 !
;   63          0063  1 !         V03-004 PCA0060          Paul Anagnostopoulos    29-Mar-1982
;   64          0064  1 !                 Changed the way the index record statistics were
;   65          0065  1 !                 calculated to make them parallel to the data record.
;   66          0066  1 !
;   67          0067  1 !         V03-003 PCA0051          Paul Anagnostopoulos    26-mar-1982
;   68          0068  1 !                 The statistics callback that specified the nominal
;   69          0069  1 !                 length of the data record did not include the key.
;   70          0070  1 !
;   71          0071  1 !         V03-002 PCA0004          Paul Anagnostopoulos    16-Mar-1982
;   72          0072  1 !                 The key significance count is no longer present in
;   73          0073  1 !                 the data bucket trailer.
;   74          0074  1 !
;   75          0075  1 !         V03-001 PCA0003          Paul Anagnostopoulos    16-Mar-1982
;   76          0076  1 !                 A bug in ANL$3RECLAIMED_BUCKET_HEADER caused it to
;   77          0077  1 !                 sometimes think the bucket header was not at the
;   78          0078  1 !                 beginning of the bucket.
;   79          0079  1 !--
;    81          0080  1 %sbttl 'Module Declarations'
;    82          0081  1 !
;    83          0082  1 ! Libraries and Requires:
;    84          0083  1 !
;    85          0084  1
;    86          0085  1 library 'lib';
;    87          0086  1 require 'rmsreq';
;    88          0595  1
;    89          0596  1 !
;    90          0597  1 ! Table of Contents:
;    91          0598  1 !
;    92          0599  1
;    93          0600  1 forward routine
;    94          0601  1         anl$3bucket_header,
;    95          0602  1         anl$3reclaimed_bucket_header,
;    96          0603  1         anl$3index_record,
;    97          0604  1         anl$3primary_data_record,
;    98          0605  1         anl$3format_data_bytes: novalue,
;    99          0606  1         calculate_data_record_info: novalue,
;   100          0607  1         anl$3sidr_record,
;   101          0608  1         anl$3sidr_pointer;
;   102          0609  1
;   103          0610  1 !
;   104          0611  1 ! External References:
;   105          0612  1 !
;   106          0613  1
;   107          0614  1 external routine
;   108          0615  1         anl$bucket,
;   109          0616  1         anl$bucket_callback,
;   110          0617  1         anl$check_flags,
;   111          0618  1         anl$data_callback,
;   112          0619  1         anl$format_error,
;   113          0620  1         anl$format_flags,
;   114          0621  1         anl$format_hex,
;   115          0622  1         anl$format_line,
;   116          0623  1         anl$format_skip,
;   117          0624  1         anl$index_callback,
;   118          0625  1         anl$reclaimed_bucket_callback;
;   119          0626  1
;   120          0627  1 external
;   121          0628  1         anl$gb_mode: byte,
;   122          0629  1         anl$gl_fat: ref block[,byte],
;   123          0630  1         anl$gw_prolog: word;
;   124          0631  1
;   125          0632  1 !
;   126          0633  1 ! Own Variables:
;   127          0634  1 !
```

```
1 %sbttl 'ANL$3BUCKET_HEADER - Print and Check a Bucket Header'
1 !++
1 ! Functional Description:
1 !       This routine is responsible for printing and checking the contents
1 !       of the bucket header in prolog 3 indexed file buckets.
1 !
1 ! Formal Parameters:
1 !       the_bsd         The address of a BSD describing the complete bucket.
1 !                       We update it to the next bucket.
1 !       key_id          The alleged ID of the key descriptor for this bucket.
1 !       dups            A boolean, true if duplicates allowed for this key.
1 !       level           The alleged level of this bucket.
1 !       report          A boolean, true if we are to print a report.
1 !       indent_level    The indentation level of the report.
1 !
1 ! Implicit Inputs:
1 !       global data
1 !
1 ! Implicit Outputs:
1 !       global data
1 !
1 ! Returned Value:
1 !       True if there is another bucket in this chain, false otherwise.
1 !
1 ! Side Effects:
1 !
1 !--
1
1
2 global routine anl$3bucket_header(the_bsd,key_id,dups,level,report,indent_level) = begin
2
2 bind
2       b = .the_bsd: bsd;
2
2 own
2       index_flags_def: block[3,long] initial(
2                               1,
2                               uplit byte (%ascic 'BKT$V_LASTBKT'),
2                               uplit byte (%ascic 'BKT$V_ROOTBKT')
2                               ),
2
2       data_flags_def: block[2,long] initial(
2                               0,
2                               uplit byte (%ascic 'BKT$V_LASTBKT')
2                               );
2
2 local
2       sp: ref block[,byte],
2       tp: ref block[,byte];
2
2 ! We know the bucket header fits in the bucket.  Set up a pointer to the header
2 ! and a pointer to the trailer, which is the last 8 bytes.
2
2 sp = .b[bsd$l_bufptr];
2 tp = .b[bsd$l_endptr] - 8;
2
```

```
;  186        0692  2 ! Now we can format the header if requested.
;  187        0693  2
;  188        0694  3 if .report then (
;  189        0695  3
;  190        0696  3         ! Start with a nice header, containing the VBN.
;  191        0697  3
;  192        0698  3         anl$format_line(3,.indent_level,anlrms$_bkt,.b[bsd$l_vbn]);
;  193        0699  3         anl$format_skip(0);
;  194        0700  3
;  195        0701  3         ! Format the check character.
;  196        0702  3
;  197        0703  3         anl$format_line(0,.indent_level+1,anlrms$_bktcheck,.sp[bkt$b_checkchar]);
;  198        0704  3
;  199        0705  3         ! Format the key ID.
;  200        0706  3
;  201        0707  3         anl$format_line(0,.indent_level+1,anlrms$_bktkey,.sp[bkt$b_indexno]);
;  202        0708  3
;  203        0709  3         ! Now the VBN address sample.
;  204        0710  3
;  205        0711  3         anl$format_line(0,.indent_level+1,anlrms$_bktsample,.sp[bkt$w_adrsample]);
;  206        0712  3
;  207        0713  3         ! Now the free space offset.
;  208        0714  3
;  209        0715  3         anl$format_line(0,.indent_level+1,anlrms$_bktfree,.sp[bkt$w_keyfrespc]);
;  210        0716  3
;  211        0717  3         ! Now the next available record ID.
;  212        0718  3
;  213        0719  3         anl$format_line(0,.indent_level+1,anlrms$_bktrecid3,.sp[bkt$w_nxtrecid]);
;  214        0720  3
;  215        0721  3         ! Now the next bucket VBN.
;  216        0722  3
;  217        0723  3         anl$format_line(0,.indent_level+1,anlrms$_bktnext,.sp[bkt$l_nxtbkt]);
;  218        0724  3
;  219        0725  3         ! Now the level number.
;  220        0726  3
;  221        0727  3         anl$format_line(0,.indent_level+1,anlrms$_bktlevel,.sp[bkt$b_level]);
;  222        0728  3
;  223        0729  3         ! Now the control bits.
;  224        0730  3
;  225        0731  3         anl$format_flags(.indent_level+1,anlrms$_bktflags,.sp[bkt$b_bktcb],
;  226        0732  3                 (if .sp[bkt$b_level] eqlu 0 then data_flags_def else index_flags_def));
;  227        0733  3
;  228        0734  3         ! Now the VBN list pointer size, but only if this is an index bucket.
;  229        0735  3
;  230        0736  3         if .sp[bkt$b_level] gtru 0 then
;  231        0737  3                 anl$format_line(0,.indent_level+1,anlrms$_bktptrsize,.sp[bkt$v_ptr_sz]+2);
;  232        0738  3
;  233        0739  3         ! Now we are going to format the stuff at the end of the bucket.
;  234        0740  3         ! There is only the VBN free space offset if this is an index bucket.
;  235        0741  3
;  236        0742  3         anl$format_skip(0);
;  237        0743  3         if .sp[bkt$b_level] gtru 0 then
;  238        0744  3                 anl$format_line(0,.indent_level+1,anlrms$_bktvbnfree,.tp[4,0,16,0]);
;  239        0745  2 );
;  241            0746  2 ! Now we are going the check the contents of the bucket header.  This is a
;  242            0747  2 ! fairly rigorous test, but doesn't check anything that requires looking
;  243            0748  2 ! at other structures.
;  244            0749  2
;  245            0750  2 ! Make sure the check byte is present in the last byte of the bucket.
;  246            0751  2
;  247            0752  2 if .sp[bkt$b_checkchar] nequ ch$rchar(.b[bsd$l_endptr]-1) then
;  248            0753  2         anl$format_error(anlrms$_badbktcheck,.b[bsd$l_vbn]);
;  249            0754  2
;  250            0755  2 ! Check the key ID.
;  251            0756  2
;  252            0757  2 if .sp[bkt$b_indexno] nequ .key_id then
;  253            0758  2         anl$format_error(anlrms$_badbktkeyid,.b[bsd$l_vbn]);
;  254            0759  2
;  255            0760  2 ! Check the bucket address sample.
;  256            0761  2
;  257            0762  2 if .sp[bkt$w_adrsample] nequ (.b[bsd$l_vbn] and %x'0000ffff') then
;  258            0763  2         anl$format_error(anlrms$_badbktsample,.b[bsd$l_vbn]);
;  259            0764  2
;  260            0765  2 ! Check that the next available byte is within reasonable limits.
;  261            0766  2 !!!TEMP!!!
;  262            0767  2
;  263            0768  2 if .sp[bkt$w_freespace] lssu bkt$c_overhdsz or
;  264            0769  2    .sp[bkt$w_freespace] gtru .b[bsd$w_size]*512-1 then
;  265            0770  2         anl$format_error(anlrms$_badbktfree,.b[bsd$l_vbn]);
;  266            0771  2
;  267            0772  2 ! Check the level number.
;  268            0773  2
;  269            0774  2 if .sp[bkt$b_level] nequ .level then
;  270            0775  2         anl$format_error(anlrms$_badbktlevel,.b[bsd$l_vbn]);
;  271            0776  2
;  272            0777  2 ! Check the byte of control flags.  Make sure we don't get confused by
;  273            0778  2 ! the pointer size.
;  274            0779  2
;  275            0780  2 anl$check_flags(.b[bsd$l_vbn],.sp[bkt$b_bktcb] and %x'e7',
;  276            0781  2                 (if .sp[bkt$b_level] eqlu 0 then data_flags_def else index_flags_def));
;  277            0782  2
;  278            0783  2 ! Now split up depending on the type of bucket.
;  279            0784  2
;  280            0785  3 if .sp[bkt$b_level] gtru 0 then (
;  281            0786  3
;  282            0787  3         ! This is an index bucket.  Check the VBN free space offset.
;  283            0788  3         ! If we are accumulating statistics, then call the bucket callback
;  284            0789  3         ! routine, telling it the level, bucket size, and fill amount.
;  285            0790  3
;  286            0791  3         if .tp[4,0,16,0] lssu .sp[bkt$w_freespace]-1 or
;  287            0792  3            .tp[4,0,16,0] gtru .b[bsd$w_size]*512-1       then
;  288            0793  3                 anl$format_error(anlrms$_badvbnfree,.b[bsd$l_vbn]);
;  289            0794  3
;  290          P 0795  3         statistics_callback(
;  291          P 0796  3                 anl$bucket_callback(.sp[bkt$b_level],
;  292          P 0797  3                                     .b[bsd$w_size],
;  293          P 0798  3                                     .b[bsd$w_size]*512 - .tp[4,0,16,0] + .sp[bkt$w_freespace] - 1);
;  294            0799  3         );
;  295            0800  3
;  296            0801  2 ) else
;  297            0802  2
;  298          0803  2           ! All we need to do for data buckets is call the statistics
;  299          0804  2           ! callback routine with the same information.
;  300          0805  2
;  301        P 0806  2           statistics_callback(
;  302        P 0807  2                   anl[$bucket_callback(.sp[bkt$b_level],
;  303        P 0808  2                                        .b[bsd$w_size],
;  304        P 0809  2                                        .sp[bkt$w_freespace] + 1);
;  305          0810  2           );
;  307         0811  2 ! If this is not the last bucket in this chain, then let's update the
;  308         0812  2 ! BSD to describe the next one.  Otherwise forget it.
;  309         0813  2
;  310         0814  3 if not .sp[bkt$v_lastbkt] then (
;  311         0815  3         b[bsd$l_vbn] = .sp[bkt$l_nxtbkt];
;  312         0816  3         anl$bucket(b,0);
;  313         0817  3         return true;
;  314         0818  2 ) else
;  315         0819  2         return false;
;  316         0820  2
;  317         0821  1 end;


                                                                    .TITLE  RMS3IDX RMS3IDX - Analyze Things for Prolog 3 I
;                                                                                   ndexed F
                                                                    .IDENT  \V04-000\

                                                                    .PSECT  $PLIT$,NOWRT,NOEXE,2

    54  4B  42  54  53  41  4C  5F  56  24  54  4B  42  0D  00000 P.AAA:  .ASCII  <13>\BKT$V_LASTBKT\                          ;
    54  4B  42  54  4F  4F  52  5F  56  24  54  4B  42  0D  0000E P.AAB:  .ASCII  <13>\BKT$V_ROOTBKT\                          ;
    54  4B  42  54  53  41  4C  5F  56  24  54  4B  42  0D  0001C P.AAC:  .ASCII  <13>\BKT$V_LASTBKT\                          ;

                                                                    .PSECT  $OWN$,NOEXE,2

                                          00000001  00000 INDEX_FLAGS_DEF:
                                                                    .LONG   1                                                  ;
                               00000000'  00000000'  00004          .ADDRESS P.AAA, P.AAB                                      ;
                                          00000000  0000C DATA_FLAGS_DEF:
                                                                    .LONG   0                                                  ;
                                          00000000'  00010          .ADDRESS P.AAC                                             ;

                                                                    .EXTRN  ANLRMS$_OK, ANLRMS$_ALLOC
                                                                    .EXTRN  ANLRMS$_ANYTHING
                                                                    .EXTRN  ANLRMS$_BACKUP, ANLRMS$_BKT
                                                                    .EXTRN  ANLRMS$_BKTAREA
                                                                    .EXTRN  ANLRMS$_BKTCHECK
                                                                    .EXTRN  ANLRMS$_BKTFLAGS
                                                                    .EXTRN  ANLRMS$_BKTFREE
                                                                    .EXTRN  ANLRMS$_BKTKEY, ANLRMS$_BKTLEVEL
                                                                    .EXTRN  ANLRMS$_BKTNEXT
                                                                    .EXTRN  ANLRMS$_BKTPTRSIZE
                                                                    .EXTRN  ANLRMS$_BKTRECID
                                                                    .EXTRN  ANLRMS$_BKTRECID3
                                                                    .EXTRN  ANLRMS$_BKTSAMPLE
                                                                    .EXTRN  ANLRMS$_BKTVBNFREE
                                                                    .EXTRN  ANLRMS$_BUCKETSIZE
                                                                    .EXTRN  ANLRMS$_CELL, ANLRMS$_CELLDATA
                                                                    .EXTRN  ANLRMS$_CELLFLAGS
                                                                    .EXTRN  ANLRMS$_CHECKHDG
                                                                    .EXTRN  ANLRMS$_CONTIG, ANLRMS$_CREATION
                                                                    .EXTRN  ANLRMS$_CTLSIZE
                                                                    .EXTRN  ANLRMS$_DATAREC
                                                                    .EXTRN  ANLRMS$_DATABKTVBN
                                                                    .EXTRN  ANLRMS$_DUMPHEADING
                                                                    .EXTRN  ANLRMS$_EOF, ANLRMS$_ERRORCOUNT
                                                                    .EXTRN  ANLRMS$_ERRORNONE
```

```
        .EXTRN  ANLRMS$_ERRORS, ANLRMS$_EXPIRATION
        .EXTRN  ANLRMS$_FILEATTR
        .EXTRN  ANLRMS$_FILEHDR
        .EXTRN  ANLRMS$_FILEID, ANLRMS$_FILEORG
        .EXTRN  ANLRMS$_FILESPEC
        .EXTRN  ANLRMS$_FLAG, ANLRMS$_GLOBALBUFS
        .EXTRN  ANLRMS$_HEXDATA
        .EXTRN  ANLRMS$_HEXHEADING1
        .EXTRN  ANLRMS$_HEXHEADING2
        .EXTRN  ANLRMS$_IDXAREA
        .EXTRN  ANLRMS$_IDXAREAALLOC
        .EXTRN  ANLRMS$_IDXAREABKTSZ
        .EXTRN  ANLRMS$_IDXAREANEXT
        .EXTRN  ANLRMS$_IDXAREANOALLOC
        .EXTRN  ANLRMS$_IDXAREAQTY
        .EXTRN  ANLRMS$_IDXAREARECL
        .EXTRN  ANLRMS$_IDXAREAUSED
        .EXTRN  ANLRMS$_IDXKEY, ANLRMS$_IDXKEYAREAS
        .EXTRN  ANLRMS$_IDXKEYBKTSZ
        .EXTRN  ANLRMS$_IDXKEYBYTES
        .EXTRN  ANLRMS$_IDXKEY1TYPE
        .EXTRN  ANLRMS$_IDXKEYDATAVBN
        .EXTRN  ANLRMS$_IDXKEYFILL
        .EXTRN  ANLRMS$_IDXKEYFLAGS
        .EXTRN  ANLRMS$_IDXKEYKEYSZ
        .EXTRN  ANLRMS$_IDXKEYNAME
        .EXTRN  ANLRMS$_IDXKEYNEXT
        .EXTRN  ANLRMS$_IDXKEYMINREC
        .EXTRN  ANLRMS$_IDXKEYNULL
        .EXTRN  ANLRMS$_IDXKEYPOSS
        .EXTRN  ANLRMS$_IDXKEYROOTLVL
        .EXTRN  ANLRMS$_IDXKEYROOTVBN
        .EXTRN  ANLRMS$_IDXKEYSEGS
        .EXTRN  ANLRMS$_IDXKEYSIZES
        .EXTRN  ANLRMS$_IDXPRIMREC
        .EXTRN  ANLRMS$_IDXPRIMRECFLAGS
        .EXTRN  ANLRMS$_IDXPRIMRECID
        .EXTRN  ANLRMS$_IDXPRIMRECLEN
        .EXTRN  ANLRMS$_IDXPRIMRECRRV
        .EXTRN  ANLRMS$_IDXPROAREAS
        .EXTRN  ANLRMS$_IDXPROLOG
        .EXTRN  ANLRMS$_IDXREC, ANLRMS$_IDXRECPTR
        .EXTRN  ANLRMS$_IDXSIDR
        .EXTRN  ANLRMS$_IDXSIDRDUPCNT
        .EXTRN  ANLRMS$_IDXSIDRFLAGS
        .EXTRN  ANLRMS$_IDXSIDRRECID
        .EXTRN  ANLRMS$_IDXSIDRPTRFLAGS
        .EXTRN  ANLRMS$_IDXSIDRPTRREF
        .EXTRN  ANLRMS$_INTERCOMMAND
        .EXTRN  ANLRMS$_INTERHDG
        .EXTRN  ANLRMS$_LONGREC
        .EXTRN  ANLRMS$_MAXRECSIZE
        .EXTRN  ANLRMS$_NOBACKUP
        .EXTRN  ANLRMS$_NOEXPIRATION
        .EXTRN  ANLRMS$_NOSPANFILLER
        .EXTRN  ANLRMS$_PERFORM
        .EXTRN  ANLRMS$_PROLOGFLAGS
```

```
.EXTRN  ANLRMS$_PROLOGVER
.EXTRN  ANLRMS$_PROT, ANLRMS$_RECATTR
.EXTRN  ANLRMS$_RECFMT, ANLRMS$_RECLAIMBKT
.EXTRN  ANLRMS$_RELBUCKET
.EXTRN  ANLRMS$_RELEOFVBN
.EXTRN  ANLRMS$_RELMAXREC
.EXTRN  ANLRMS$_RELPROLOG
.EXTRN  ANLRMS$_RELIAB, ANLRMS$_REVISION
.EXTRN  ANLRMS$_STATHDG
.EXTRN  ANLRMS$_SUMMARYHDG
.EXTRN  ANLRMS$_OWNERUIC
.EXTRN  ANLRMS$_JNL, ANLRMS$_AIJNL
.EXTRN  ANLRMS$_BIJNL, ANLRMS$_ATJNL
.EXTRN  ANLRMS$_ATTOP, ANLRMS$_BADCMD
.EXTRN  ANLRMS$_BADPATH
.EXTRN  ANLRMS$_BADVBN, ANLRMS$_DOWNHELP
.EXTRN  ANLRMS$_DOWNPATH
.EXTRN  ANLRMS$_EMPTYBKT
.EXTRN  ANLRMS$_NODATA, ANLRMS$_NODOWN
.EXTRN  ANLRMS$_NONEXT, ANLRMS$_NORECLAIMED
.EXTRN  ANLRMS$_NORECS, ANLRMS$_NORRV
.EXTRN  ANLRMS$_RESTDONE
.EXTRN  ANLRMS$_STACKFULL
.EXTRN  ANLRMS$_UNINITINDEX
.EXTRN  ANLRMS$_FDLIDENT
.EXTRN  ANLRMS$_FDLSYSTEM
.EXTRN  ANLRMS$_FDLSOURCE
.EXTRN  ANLRMS$_FDLFILE
.EXTRN  ANLRMS$_FDLALLOC
.EXTRN  ANLRMS$_FDLNOALLOC
.EXTRN  ANLRMS$_FDLBESTTRY
.EXTRN  ANLRMS$_FDLBUCKETSIZE
.EXTRN  ANLRMS$_FDLCLUSTERSIZE
.EXTRN  ANLRMS$_FDLCONTIG
.EXTRN  ANLRMS$_FDLEXTENSION
.EXTRN  ANLRMS$_FDLGLOBALBUFS
.EXTRN  ANLRMS$_FDLMAXRECORD
.EXTRN  ANLRMS$_FDLFILENAME
.EXTRN  ANLRMS$_FDLORG, ANLRMS$_FDLOWNER
.EXTRN  ANLRMS$_FDLPROTECTION
.EXTRN  ANLRMS$_FDLRECORD
.EXTRN  ANLRMS$_FDLSPAN
.EXTRN  ANLRMS$_FDLCC, ANLRMS$_FDLVFCSIZE
.EXTRN  ANLRMS$_FDLFORMAT
.EXTRN  ANLRMS$_FDLSIZE
.EXTRN  ANLRMS$_FDLAREA
.EXTRN  ANLRMS$_FDLKEY, ANLRMS$_FDLCHANGES
.EXTRN  ANLRMS$_FDLDATAAREA
.EXTRN  ANLRMS$_FDLDATAFILL
.EXTRN  ANLRMS$_FDLDATAKEYCOMPB
.EXTRN  ANLRMS$_FDLDATARECCOMPB
.EXTRN  ANLRMS$_FDLDUPS
.EXTRN  ANLRMS$_FDLINDEXAREA
.EXTRN  ANLRMS$_FDLINDEXCOMPB
.EXTRN  ANLRMS$_FDLINDEXFILL
.EXTRN  ANLRMS$_FDLL1INDEXAREA
.EXTRN  ANLRMS$_FDLKEYNAME
```

```
        .EXTRN  ANLRMS$_FDLNORECS
        .EXTRN  ANLRMS$_FDLNULLKEY
        .EXTRN  ANLRMS$_FDLNULLVALUE
        .EXTRN  ANLRMS$_FDLPROLOG
        .EXTRN  ANLRMS$_FDLSEGLENGTH
        .EXTRN  ANLRMS$_FDLSEGPOS
        .EXTRN  ANLRMS$_FDLSEGTYPE
        .EXTRN  ANLRMS$_FDLANALAREA
        .EXTRN  ANLRMS$_FDLRECL
        .EXTRN  ANLRMS$_FDLANALKEY
        .EXTRN  ANLRMS$_FDLDATAKEYCOMP
        .EXTRN  ANLRMS$_FDLDATARECCOMP
        .EXTRN  ANLRMS$_FDLDATARECS
        .EXTRN  ANLRMS$_FDLDATASPACE
        .EXTRN  ANLRMS$_FDLDEPTH
        .EXTRN  ANLRMS$_FDLDUPSPER
        .EXTRN  ANLRMS$_FDLIDXCOMP
        .EXTRN  ANLRMS$_FDLIDXFILL
        .EXTRN  ANLRMS$_FDLIDXSPACE
        .EXTRN  ANLRMS$_FDLIDXL1RECS
        .EXTRN  ANLRMS$_FDLDATALENMEAN
        .EXTRN  ANLRMS$_FDLIDXLENMEAN
        .EXTRN  ANLRMS$_STATAREA
        .EXTRN  ANLRMS$_STATRECL
        .EXTRN  ANLRMS$_STATKEY
        .EXTRN  ANLRMS$_STATDEPTH
        .EXTRN  ANLRMS$_STATIDXL1RECS
        .EXTRN  ANLRMS$_STATIDXLENMEAN
        .EXTRN  ANLRMS$_STATIDXSPACE
        .EXTRN  ANLRMS$_STATIDXFILL
        .EXTRN  ANLRMS$_STATIDXCOMP
        .EXTRN  ANLRMS$_STATDATARECS
        .EXTRN  ANLRMS$_STATDUPSPER
        .EXTRN  ANLRMS$_STATDATALENMEAN
        .EXTRN  ANLRMS$_STATDATASPACE
        .EXTRN  ANLRMS$_STATDATAFILL
        .EXTRN  ANLRMS$_STATDATAKEYCOMP
        .EXTRN  ANLRMS$_STATDATARECCOMP
        .EXTRN  ANLRMS$_STATEFFICIENCY
        .EXTRN  ANLRMS$_BADAREA1ST2
        .EXTRN  ANLRMS$_BADAREABKTSIZE
        .EXTRN  ANLRMS$_BADAREAFIT
        .EXTRN  ANLRMS$_BADAREAID
        .EXTRN  ANLRMS$_BADAREANEXT
        .EXTRN  ANLRMS$_BADAREAROOT
        .EXTRN  ANLRMS$_BADAREAUSED
        .EXTRN  ANLRMS$_BADBKTAREAID
        .EXTRN  ANLRMS$_BADBKTCHECK
        .EXTRN  ANLRMS$_BADBKTFREE
        .EXTRN  ANLRMS$_BADBKTKEYID
        .EXTRN  ANLRMS$_BADBKTLEVEL
        .EXTRN  ANLRMS$_BADBKTROOTBIT
        .EXTRN  ANLRMS$_BADBKTSAMPLE
        .EXTRN  ANLRMS$_BADCELLFIT
        .EXTRN  ANLRMS$_BADCHECKSUM
        .EXTRN  ANLRMS$_BADDATARECBITS
        .EXTRN  ANLRMS$_BADDATARECFIT
```

```
                                                        .EXTRN  ANLRMS$_BADDATARECPS
                                                        .EXTRN  ANLRMS$_BAD3IDXKEYFIT
                                                        .EXTRN  ANLRMS$_BADIDXLASTKEY
                                                        .EXTRN  ANLRMS$_BADIDXORDER
                                                        .EXTRN  ANLRMS$_BADIDXRECBITS
                                                        .EXTRN  ANLRMS$_BADIDXRECFIT
                                                        .EXTRN  ANLRMS$_BADIDXRECPS
                                                        .EXTRN  ANLRMS$_BADKEYAREAID
                                                        .EXTRN  ANLRMS$_BADKEYDATABKT
                                                        .EXTRN  ANLRMS$_BADKEYDATAFIT
                                                        .EXTRN  ANLRMS$_BADKEYDATATYPE
                                                        .EXTRN  ANLRMS$_BADKEYIDXBKT
                                                        .EXTRN  ANLRMS$_BADKEYFILL
                                                        .EXTRN  ANLRMS$_BADKEYFIT
                                                        .EXTRN  ANLRMS$_BADKEYREFID
                                                        .EXTRN  ANLRMS$_BADKEYROOTLEVEL
                                                        .EXTRN  ANLRMS$_BADKEYSEGCOUNT
                                                        .EXTRN  ANLRMS$_BADKEYSEGVEC
                                                        .EXTRN  ANLRMS$_BADKEYSUMMARY
                                                        .EXTRN  ANLRMS$_BADREADNOPAR
                                                        .EXTRN  ANLRMS$_BADREADPAR
                                                        .EXTRN  ANLRMS$_BADSIDRDUPCT
                                                        .EXTRN  ANLRMS$_BADSIDRPTRFIT
                                                        .EXTRN  ANLRMS$_BADSIDRPTRSZ
                                                        .EXTRN  ANLRMS$_BADSIDRSIZE
                                                        .EXTRN  ANLRMS$_BADSTREAMEOF
                                                        .EXTRN  ANLRMS$_BADVBNFREE
                                                        .EXTRN  ANLRMS$_BKTLOOP
                                                        .EXTRN  ANLRMS$_EXTENDERR
                                                        .EXTRN  ANLRMS$_FLAGERROR
                                                        .EXTRN  ANLRMS$_MISSINGBKT
                                                        .EXTRN  ANLRMS$_NOTOK, ANLRMS$_SPANERROR
                                                        .EXTRN  ANLRMS$_TOOMANYRECS
                                                        .EXTRN  ANLRMS$_UNWIND, ANLRMS$_VFCTOOSHORT
                                                        .EXTRN  ANLRMS$_CACHEFULL
                                                        .EXTRN  ANLRMS$_CACHERELFAIL
                                                        .EXTRN  ANLRMS$_FACILITY
                                                        .EXTRN  ANL$BUCKET, ANL$BUCKET_CALLBACK
                                                        .EXTRN  ANL$CHECK_FLAGS
                                                        .EXTRN  ANL$DATA_CALLBACK
                                                        .EXTRN  ANL$FORMAT_ERROR
                                                        .EXTRN  ANL$FORMAT_FLAGS
                                                        .EXTRN  ANL$FORMAT_HEX, ANL$FORMAT_LINE
                                                        .EXTRN  ANL$FORMAT_SKIP
                                                        .EXTRN  ANL$INDEX_CALLBACK
                                                        .EXTRN  ANL$RECLAIMED_BUCKET_CALLBACK
                                                        .EXTRN  ANL$GB_MODE, ANL$GL_FAT
                                                        .EXTRN  ANL$GW_PROLOG

                                                        .PSECT  $CODE$,NOWRT,2

                          0FFC 00000                    .ENTRY  ANL$3BUCKET_HEADER, Save R2,R3,R4,R5,R6,R7,-; 0664
                                                                R8,R9,R10,R11
5B       0000G   CF    9E 00002                         MOVAB   ANL$GB_MODE, R11
5A       0000'   CF    9E 00007                         MOVAB   DATA_FLAGS_DEF, R10
59       0000G   CF    9E 0000C                         MOVAB   ANL$FORMAT_ERROR, R9
58       0000G   CF    9E 00011                         MOVAB   ANL$FORMAT_LINE, R8
```

```
                54          04   AC D0 00016          MOVL     THE BSD, R4                      ; 0667
                53          0C   A4 D0 0001A          MOVL     12(R4), SP                       ; 0689
                56          10   A4 D0 0001E          MOVL     16(R4), R6                       ; 0690
                52          F8   A6 9E 00022          MOVAB    -8(R6), TP                       :
                03          14   AC E8 00026          BLBS     REPORT, 1$                       ; 0694
                              00ED 31 0002A          BRW      5$                               :
                            04   A4 DD 0002D 1$:      PUSHL    4(R4)                            ; 0698
                     00000000G   8F DD 00030          PUSHL    #ANLRMS$ BKT                     :
                            18   AC DD 00036          PUSHL    INDENT_LEVEL                     :
                                 03 DD 00039          PUSHL    #3                               :
                68               04 FB 0003B          CALLS    #4, ANL$FORMAT_LINE              :
                                 7E D4 0003E          CLRL     -(SP)                            ; 0699
         0000G  CF               01 FB 00040          CALLS    #1, ANL$FORMAT_SKIP              :
                7E               63 9A 00045          MOVZBL   (SP), -(SP)                      ; 0703
                     00000000G   8F DD 00048          PUSHL    #ANLRMS$_BKTCHECK                :
55          18  AC               01 C1 0004E          ADDL3    #1, INDENT_LEVEL, R5             :
                                 55 DD 00053          PUSHL    R5                               :
                                 7E D4 00055          CLRL     -(SP)                            :
                68               04 FB 00057          CALLS    #4, ANL$FORMAT_LINE              :
                7E          01   A3 9A 0005A          MOVZBL   1(SP), -(SP)                     ; 0707
                     00000000G   8F DD 0005E          PUSHL    #ANLRMS$_BKTKEY                  :
                                 55 DD 00064          PUSHL    R5                               :
                                 7E D4 00066          CLRL     -(SP)                            :
                68               04 FB 00068          CALLS    #4, ANL$FORMAT_LINE              :
                7E          02   A3 3C 0006B          MOVZWL   2(SP), -(SP)                     ; 0711
                     00000000G   8F DD 0006F          PUSHL    #ANLRMS$_BKTSAMPLE               :
                                 55 DD 00075          PUSHL    R5                               :
                                 7E D4 00077          CLRL     -(SP)                            :
                68               04 FB 00079          CALLS    #4, ANL$FORMAT_LINE              :
                7E          04   A3 3C 0007C          MOVZWL   4(SP), -(SP)                     ; 0715
                     00000000G   8F DD 00080          PUSHL    #ANLRMS$_BKTFREE                 :
                                 55 DD 00086          PUSHL    R5                               :
                                 7E D4 00088          CLRL     -(SP)                            :
                68               04 FB 0008A          CALLS    #4, ANL$FORMAT_LINE              :
                7E          06   A3 3C 0008D          MOVZWL   6(SP), -(SP)                     ; 0719
                     00000000G   8F DD 00091          PUSHL    #ANLRMS$_BKTRECID3               :
                                 55 DD 00097          PUSHL    R5                               :
                                 7E D4 00099          CLRL     -(SP)                            :
                68               04 FB 0009B          CALLS    #4, ANL$FORMAT_LINE              :
                            08   A3 DD 0009E          PUSHL    8(SP)                            ; 0723
                     00000000G   8F DD 000A1          PUSHL    #ANLRMS$_BKTNEXT                 :
                                 55 DD 000A7          PUSHL    R5                               :
                                 7E D4 000A9          CLRL     -(SP)                            :
                68               04 FB 000AB          CALLS    #4, ANL$FORMAT_LINE              :
                7E          0C   A3 9A 000AE          MOVZBL   12(SP), -(SP)                    ; 0727
                     00000000G   8F DD 000B2          PUSHL    #ANLRMS$_BKTLEVEL                :
                                 55 DD 000B8          PUSHL    R5                               :
                                 7E D4 000BA          CLRL     -(SP)                            :
                68               04 FB 000BC          CALLS    #4, ANL$FORMAT_LINE              :
                            0C   A3 95 000BF          TSTB     12(SP)                           ; 0732
                                 05 12 000C2          BNEQ     2$                               :
                50               6A 9E 000C4          MOVAB    DATA_FLAGS_DEF, R0               :
                                 04 11 000C7          BRB      3$                               :
                50          F4   AA 9E 000C9 2$:      MOVAB    INDEX_FLAGS_DEF, R0              :
                                 50 DD 000CD 3$:      PUSHL    R0                               :
                7E          0D   A3 9A 000CF          MOVZBL   13(SP), -(SP)                    ; 0731
                     00000000G   8F DD 000D3          PUSHL    #ANLRMS$_BKTFLAGS                :
```

```
                                                 55  DD 000D9            PUSHL   R5                            :
                           0000G  CF             04  FB 000DB            CALLS   #4, ANL$FORMAT_FLAGS          :
                                                 57  D4 000E0            CLRL    R7                            : 0736
                                             0C  A3  95 000E2            TSTB    12(SP)                        :
                                                 18  13 000E5            BEQL    4$                            :
                                                 57  D6 000E7            INCL    R7                            :
      7E      0D    A3            02             03  EF 000E9            EXTZV   #3, #2, 13(SP), -(SP)         : 0737
                                  6E             02  C0 000EF            ADDL2   #2, (SP)                      :
                                      00000000G  8F  DD 000F2            PUSHL   #ANLRMS$_BKTPTRSIZE           :
                                                 55  DD 000F8            PUSHL   R5                            :
                                                 7E  D4 000FA            CLRL    -(SP)                         :
                                  68             04  FB 000FC            CALLS   #4, ANL$FORMAT_LINE           :
                                                 7E  D4 000FF 4$:        CLRL    -(SP)                         : 0742
                           0000G  CF             01  FB 00101            CALLS   #1, ANL$FORMAT_SKIP           :
                                  11             57  E9 00106            BLBC    R7, 5$                        : 0743
                                  7E         04  A2  3C 00109            MOVZWL  4(TP), -(SP)                  : 0744
                                      00000000G  8F  DD 0010D            PUSHL   #ANLRMS$_BKTVBNFREE           :
                                                 55  DD 00113            PUSHL   R5                            :
                                                 7E  D4 00115            CLRL    -(SP)                         :
                                  68             04  FB 00117            CALLS   #4, ANL$FORMAT_LINE           :
                              FF  A6             63  91 0011A 5$:        CMPB    (SP), -1(R6)                  : 0752
                                                 0C  13 0011E            BEQL    6$                            :
                                             04  A4  DD 00120            PUSHL   4(R4)                         : 0753
                                      00000000G  8F  DD 00123            PUSHL   #ANLRMS$_BADBKTCHECK          :
                                  69             02  FB 00129            CALLS   #2, ANL$FORMAT_ERROR          :
08    AC      01    A3            08             00  ED 0012C 6$:        CMPZV   #0, #8, 1(SP), KEY_ID         : 0757
                                                 0C  13 00133            BEQL    7$                            :
                                             04  A4  DD 00135            PUSHL   4(R4)                         : 0758
                                      00000000G  8F  DD 00138            PUSHL   #ANLRMS$_BADBKTKEYID          :
                                  69             02  FB 0013E            CALLS   #2, ANL$FORMAT_ERROR          :
                                  56         04  A4  D0 00141 7$:        MOVL    4(R4), R6                     : 0762
                                  56         02  A3  B1 00145            CMPW    2(SP), R6                     :
                                                 0B  13 00149            BEQL    8$                            :
                                                 56  DD 0014B            PUSHL   R6                            : 0763
                                      00000000G  8F  DD 0014D            PUSHL   #ANLRMS$_BADBKTSAMPLE         :
                                  69             02  FB 00153            CALLS   #2, ANL$FORMAT_ERROR          :
                                  55         04  A3  3C 00156 8$:        MOVZWL  4(SP), R5                     : 0768
                                  0E             55  B1 0015A            CMPW    R5, #14                       :
                                                 0F  1F 0015D            BLSSU   9$                            :
                                  50         02  A4  3C 0015F            MOVZWL  2(R4), R0                     : 0769
                    50            50             09  78 00163            ASHL    #9, R0, R0                    :
                                                 50  D7 00167            DECL    R0                            :
                                  50             55  D1 00169            CMPL    R5, R0                        :
                                                 0B  1B 0016C            BLEQU   10$                           :
                                                 56  DD 0016E 9$:        PUSHL   R6                            : 0770
                                      00000000G  8F  DD 00170            PUSHL   #ANLRMS$_BADBKTFREE           :
                                  69             02  FB 00176            CALLS   #2, ANL$FORMAT_ERROR          :
                                  57         0C  A3  9A 00179 10$:       MOVZBL  12(SP), R7                    : 0774
                              10  AC             57  D1 0017D            CMPL    R7, LEVEL                     :
                                                 0B  13 00181            BEQL    11$                           :
                                                 56  DD 00183            PUSHL   R6                            : 0775
                                      00000000G  8F  DD 00185            PUSHL   #ANLRMS$_BADBKTLEVEL          :
                                  69             02  FB 0018B            CALLS   #2, ANL$FORMAT_ERROR          :
                                                 57  D5 0018E 11$:       TSTL    R7                            : 0781
                                                 05  12 00190            BNEQ    12$                           :
                                  50             6A  9E 00192            MOVAB   DATA_FLAGS_DEF, R0            :
                                                 04  11 00195            BRB     13$                           :
```

```
                                      50       F4   AA 9E 00197 12$:     MOVAB    INDEX_FLAGS_DEF, R0           ;
                                                    50 DD 0019B 13$:     PUSHL    R0                            ;
                                      50       0D   A3 9A 0019D          MOVZBL   13(SP), R0                    ; 0780
                         7E           50 FFFFFF18   8F CB 001A1          BICL3    #-232, R0, -(SP)              ;
                                                    56 DD 001A9          PUSHL    R6                            ;
                               0000G  CF            03 FB 001AB          CALLS    #3, ANL$CHECK_FLAGS           ;
                                                    57 D5 001B0          TSTL     R7                            ; 0785
                                                    48 13 001B2          BEQL     17$                           ;
                                      50       FF   A5 9E 001B4          MOVAB    -1(R5), R0                    ; 0791
             50     04   A2           10            00 ED 001B8          CMPZV    #0, #16, 4(TP), R0            ;
                                                    12 1F 001BE          BLSSU    14$                           ;
                                      50       02   A4 3C 001C0          MOVZWL   2(R4), R0                     ; 0792
                         50           50            09 78 001C4          ASHL     #9, R0, R0                    ;
                                                    50 D7 001C8          DECL     R0                            ;
             50     04   A2           10            00 ED 001CA          CMPZV    #0, #16, 4(TP), R0            ;
                                                    0B 1B 001D0          BLEQU    15$                           ;
                                                    56 DD 001D2 14$:     PUSHL    R6                            ; 0793
                                         000000000G 8F DD 001D4          PUSHL    #ANLRMS$_BADVBNFREE           ;
                                      69            02 FB 001DA          CALLS    #2, ANL$FORMAT_ERROR          ;
                                      02            6B 91 001DD 15$:     CMPB     ANL$GB_MODE, #2               ; 0799
                                                    05 13 001E0          BEQL     16$                           ;
                                      04            6B 91 001E2          CMPB     ANL$GB_MODE, #4               ;
                                                    2D 12 001E5          BNEQ     20$                           ;
                                      50       02   A4 3C 001E7 16$:     MOVZWL   2(R4), R0                     ;
                         50           50            09 78 001EB          ASHL     #9, R0, R0                    ;
                                      51       04   A2 3C 001EF          MOVZWL   4(TP), R1                     ;
                                      50            51 C2 001F3          SUBL2    R1, R0                        ;
                                               FF A540 9F 001F6          PUSHAB   -1(R5)[R0]                    ;
                                                    0D 11 001FA          BRB      19$                           ;
                                      02            6B 91 001FC 17$:     CMPB     ANL$GB_MODE, #2               ; 0810
                                                    05 13 001FF          BEQL     18$                           ;
                                      04            6B 91 00201          CMPB     ANL$GB_MODE, #4               ;
                                                    0E 12 00204          BNEQ     20$                           ;
                                               01   A5 9F 00206 18$:     PUSHAB   1(R5)                         ;
                                      7E       02   A4 3C 00209 19$:     MOVZWL   2(R4), -(SP)                  ;
                                                    57 DD 0020D          PUSHL    R7                            ;
                               0000G  CF            03 FB 0020F          CALLS    #3, ANL$BUCKET_CALLBACK       ;
                                      12       0D   A3 E8 00214 20$:     BLBS     13(SP), 21$                   ; 0814
                                  04  A4       08   A3 D0 00218          MOVL     8(SP), 4(R4)                  ; 0815
                                                    7E D4 0021D          CLRL     -(SP)                         ; 0816
                                                    54 DD 0021F          PUSHL    R4                            ;
                               0000G  CF            02 FB 00221          CALLS    #2, ANL$BUCKET                ;
                                      50            01 D0 00226          MOVL     #1, R0                        ; 0819
                                                       04 00229          RET                                    ;
                                                    50 D4 0022A 21$:     CLRL     R0                            ;
                                                       04 0022C          RET                                    ; 0821

; Routine Size:  557 bytes,    Routine Base:  $CODE$ + 0000
```

```
%sbttl 'ANL$3RECLAIMED_BUCKET_HEADER - Check & Format Reclaimed Bucket'
!++
! Functional Description:
!       This routine is called to check and optionally format the header
!       of a reclaimed bucket.  These buckets reside on the available
!       list chained off the area descriptor.
!
! Formal Parameters:
!       the_bsd         Address of BSD describing bucket.
!       report          A boolean, true if we are to format the header.
!       indent_level    Indentation level for the report.
!
! Implicit Inputs:
!       global data
!
! Implicit Outputs:
!       global data
!
! Returned Value:
!       True if there is another bucket in the chain, false otherwise.
!
! Side Effects:
!
!--


global routine anl$3reclaimed_bucket_header(the_bsd,report,indent_level) = begin

bind
        b = .the_bsd: bsd;

own
        control_flags_def: block[2,long] initial(
                                        0,
                                        uplit byte (%ascic 'BKT$V_LASTBKT')
                                        );

local
        sp: ref block[,byte];


! We know the bucket header fits in the bucket.

! Now we can format the header if requested.

sp = .b[bsd$l_bufptr];

if .report then (

        ! Start with a nice header, containing the VBN.

        anl$format_line(3,.indent_level,anlrms$_reclaimbkt,.b[bsd$l_vbn]);
        anl$format_skip(0);

        ! Format the check character.

        anl$format_line(0,.indent_level+1,anlrms$_bktcheck,.sp[bkt$b_checkchar]);
;  376          0879  3
;  377          0880  3             ! Format the VBN address sample.
;  378          0881  3
;  379          0882  3             anl$format_line(0,.indent_level+1,anlrms$_bktsample,.sp[bkt$w_adrsample]);
;  380          0883  3
;  381          0884  3             ! Now the next available record ID.
;  382          0885  3
;  383          0886  3             anl$format_line(0,.indent_level+1,anlrms$_bktrecid3,.sp[bkt$w_nxtrecid]);
;  384          0887  3
;  385          0888  3             ! Now the next bucket VBN.
;  386          0889  3
;  387          0890  3             anl$format_line(0,.indent_level+1,anlrms$_bktnext,.sp[bkt$l_nxtbkt]);
;  388          0891  3
;  389          0892  3             ! Finally, the flags.
;  390          0893  3
;  391          0894  3             anl$format_flags(.indent_level+1,anlrms$_bktflags,.sp[bkt$b_bktcb],control_flags_def);
;  392          0895  2 );
;      2 ! Now we are going to check those items which we formatted above.  The rest
;      2 ! of the bucket header (and trailer, if prolog 3) were probably left alone
;      2 ! when the bucket was reclaimed, but we don't care.
;      2
;      2 ! Make sure the check byte is present in the last byte of the bucket.
;      2
;      2 if .sp[bkt$b_checkchar] nequ ch$rchar(.b[bsd$l_endptr]-1) then
;      2        anl$format_error(anlrms$_badbktcheck,.b[bsd$l_vbn]);
;      2
;      2 ! Check the bucket address sample.
;      2
;      2 if .sp[bkt$w_adrsample] nequ (.b[bsd$l_vbn] and %x'0000ffff') then
;      2        anl$format_error(anlrms$_badbktsample,.b[bsd$l_vbn]);
;      2
;      2 ! We can't check anything else in the header because we don't know what's
;      2 ! left over from the original bucket.
;      2
;    P 2 statistics_callback(
;    P 2
;    P 2        ! If we are accumulating statistics, then we have to call the
;    P 2        ! bucket callback routine so it can tally the bucket.
;    P 2
;    P 2        anl$reclaimed_bucket_callback(.b[bsd$w_size]);
;      2 );
```

```
2 ! If this is not the last bucket in this chain, then let's update the
2 ! BSD to describe the next one.  Otherwise forget it.
2
3 if not .sp[bkt$v_lastbkt] then (
3         b[bsd$l_vbn] = .sp[bkt$l_nxtbkt];
3         anl$bucket(b,0);
3         return true;
2 ) else
2         return false;
2
1 end;



                                                                      .PSECT  $PLIT$,NOWRT,NOEXE,2

   54  4B  42  54  53  41  4C  5F  56  24  54  4B  42  0D  0002A P.AAD:  .ASCII  <13>\BKT$V_LASTBKT\                        ;

                                                                      .PSECT  $OWN$,NOEXE,2

                                       00000000  00014 CONTROL_FLAGS_DEF:
                                                                      .LONG    0                                              ;
                                       00000000' 00018                .ADDRESS P.AAD                                          ;


                                                                      .PSECT  $CODE$,NOWRT,2

                                           003C 00000                 .ENTRY  ANL$3RECLAIMED_BUCKET_HEADER, Save R2,R3,-     ; 0848
                                                                              R4,R5                                          ;
                        55      0000G  CF   9E 00002                  MOVAB   ANL$FORMAT_LINE, R5                            ;
                        52         04  AC   D0 00007                  MOVL    THE_BSD, R2                                    ; 0851
                        53         0C  A2   D0 0000B                  MOVL    12(R2), SP                                     ; 0867
                        74         08  AC   E9 0000F                  BLBC    REPORT, 1$                                     ; 0869
                                   04  A2   DD 00013                  PUSHL   4(R2)                                          ; 0873
                            00000000G  8F   DD 00016                  PUSHL   #ANLRMS$_RECLAIMBKT                            ;
                                   0C  AC   DD 0001C                  PUSHL   INDENT_LEVEL                                   ;
                                       03   DD 0001F                  PUSHL   #3                                             ;
                        65             04   FB 00021                  CALLS   #4, ANL$FORMAT_LINE                            ;
                                       7E   D4 00024                  CLRL    -(SP)                                          ; 0874
                 0000G  CF             01   FB 00026                  CALLS   #1, ANL$FORMAT_SKIP                            ;
                        7E             63   9A 0002B                  MOVZBL  (SP), -(SP)                                    ; 0878
                            00000000G  8F   DD 0002E                  PUSHL   #ANLRMS$_BKTCHECK                              ;
      54       0C       AC             01   C1 00034                  ADDL3   #1, INDENT_LEVEL, R4                           ;
                                       54   DD 00039                  PUSHL   R4                                             ;
                                       7E   D4 0003B                  CLRL    -(SP)                                          ;
                        65             04   FB 0003D                  CALLS   #4, ANL$FORMAT_LINE                            ;
                        7E         02  A3   3C 00040                  MOVZWL  2(SP), -(SP)                                   ; 0882
                            00000000G  8F   DD 00044                  PUSHL   #ANLRMS$_BKTSAMPLE                             ;
                                       54   DD 0004A                  PUSHL   R4                                             ;
                                       7E   D4 0004C                  CLRL    -(SP)                                          ;
                        65             04   FB 0004E                  CALLS   #4, ANL$FORMAT_LINE                            ;
                        7E         06  A3   3C 00051                  MOVZWL  6(SP), -(SP)                                   ; 0886
                            00000000G  8F   DD 00055                  PUSHL   #ANLRMS$_BKTRECID3                             ;
                                       54   DD 0005B                  PUSHL   R4                                             ;
                                       7E   D4 0005D                  CLRL    -(SP)                                          ;
                        65             04   FB 0005F                  CALLS   #4, ANL$FORMAT_LINE                            ;
```

```
                                  08  A3  DD 00062          PUSHL   8(SP)                                        ; 0890
                           00000000G  8F  DD 00065          PUSHL   #ANLRMS$_BKTNEXT                             ;
                                      54  DD 0006B          PUSHL   R4                                           ;
                                      7E  D4 0006D          CLRL    -(SP)                                        ;
                      65              04  FB 0006F          CALLS   #4, ANL$FORMAT_LINE                          ;
                               0000'  CF  9F 00072          PUSHAB  CONTROL_FLAGS_DEF                            ; 0894
                      7E          0D  A3  9A 00076          MOVZBL  13(SP), -(SP)                                ;
                           00000000G  8F  DD 0007A          PUSHL   #ANLRMS$_BKTFLAGS                            ;
                                      54  DD 00080          PUSHL   R4                                           ;
              0000G   CF              04  FB 00082          CALLS   #4, ANL$FORMAT_FLAGS                         ;
                      50          10  A2  D0 00087 1$:      MOVL    16(R2), R0                                   ; 0902
                 FF   A0              63  91 0008B          CMPB    (SP), -1(R0)                                 ;
                                      0E  13 0008F          BEQL    2$                                           ;
                                  04  A2  DD 00091          PUSHL   4(R2)                                        ; 0903
                           00000000G  8F  DD 00094          PUSHL   #ANLRMS$_BADBKTCHECK                         ;
              0000G   CF              02  FB 0009A          CALLS   #2, ANL$FORMAT_ERROR                         ;
                 04   A2          02  A3  B1 0009F 2$:      CMPW    2(SP), 4(R2)                                 ; 0907
                                      0E  13 000A4          BEQL    3$                                           ;
                                  04  A2  DD 000A6          PUSHL   4(R2)                                        ; 0908
                           00000000G  8F  DD 000A9          PUSHL   #ANLRMS$_BADBKTSAMPLE                        ;
              0000G   CF              02  FB 000AF          CALLS   #2, ANL$FORMAT_ERROR                         ;
                      02       0000G  CF  91 000B4 3$:      CMPB    ANL$GB_MODE, #2                              ; 0919
                                      07  13 000B9          BEQL    4$                                           ;
                      04       0000G  CF  91 000BB          CMPB    ANL$GB_MODE, #4                              ;
                                      09  12 000C0          BNEQ    5$                                           ;
                      7E          02  A2  3C 000C2 4$:      MOVZWL  2(R2), -(SP)                                 ;
              0000G   CF              01  FB 000C6          CALLS   #1, ANL$RECLAIMED_BUCKET_CALLBACK            ;
                      12          0D  A3  E8 000CB 5$:      BLBS    13(SP), 6$                                   ; 0923
                 04   A2          08  A3  D0 000CF          MOVL    8(SP), 4(R2)                                 ; 0924
                                      7E  D4 000D4          CLRL    -(SP)                                        ; 0925
                                      52  DD 000D6          PUSHL   R2                                           ;
              0000G   CF              02  FB 000D8          CALLS   #2, ANL$BUCKET                               ;
                      50              01  D0 000DD          MOVL    #1, R0                                       ; 0928
                                          04 000E0          RET                                                  ;
                                      50  D4 000E1 6$:      CLRL    R0                                           ;
                                          04 000E3          RET                                                  ; 0930

; Routine Size:  228 bytes,    Routine Base:  $CODE$ + 022D
```

```
1 %sbttl 'ANL$3INDEX_RECORD - Format and Check an Index Record'
1 !++
1 ! Functional Description:
1 !       This routine is responsible for formatting and checking the contents
1 !       of an index record (for prolog 3).
1 !
1 ! Formal Parameters:
1 !       rec_bsd         Address of BSD describing index record.  We update it
1 !                       to describe the next record.  The work longword is
1 !                       assumed to specify the number of the record.
1 !       key_bsd         Address of BSD for key descriptor of this index.
1 !       report          A boolean, true if we are to format the record.
1 !       indent_level    Indentation level for the report.
1 !
1 ! Implicit Inputs:
1 !       global data
1 !
1 ! Implicit Outputs:
1 !       global data
1 !
1 ! Returned Value:
1 !       True if there is another index record, false otherwise.
1 !
1 ! Side Effects:
1 !
1 !--
1
1
2 global routine anl$3index_record(rec_bsd,key_bsd,report,indent_level) = begin
2
2 bind
2       b = .rec_bsd: bsd,
2       k = .key_bsd: bsd;
2
2 local
2       sp: ref block[,byte],
2       hp: ref block[,byte],
2       kp: ref block[,byte],
2       vp: ref block[,byte],
2       key_length: long;
2
2
2 ! We want to ensure that the key portion of the index record fits in the
2 ! record free space.  Begin by calculating the length of the key, which
2 ! depends on whether or not it's compressed.
2
2 hp = .b[bsd$l_bufptr];
2 sp = .b[bsd$l_bufptr] + .b[bsd$l_offset];
2 kp = .k[bsd$l_bufptr] + .k[bsd$l_offset];
2
3 key_length =    (if .kp[key$v_idx_compr] then
3                       .sp[0,0,8,0] + irc$c_keycmpovh
3               else
2                       .kp[key$b_keysz]);
2
2 ! Make sure that the key fits in the record free space.
2
```

```
;  488          0988  3 if .b[bsd$l_offset]+.key_length gtru .hp[bkt$w_keyfrespc] then (
;  489          0989  3         anl$format_error(anlrms$_bad3idxkeyfit,.b[bsd$l_vbn]);
;  490          0990  3         signal (anlrms$_unwind);
;  491          0991  2 );
;  492          0992  2
;  493          0993  2 ! Now we have to calculate the address of the corresponding VBN in the
;  494          0994  2 ! VBN list.
;  495          0995  2
;  496          0996  2 vp =    (.b[bsd$l_endptr]-4) - (.b[bsd$l_work]+1) * (.hp[bkt$v_ptr_sz]+2);
```

```
2 ! Now we can format the index record, if requested.
2
3 if .report then (
3
3         ! Begin witn a nice heading.
3
3         anl$format_l ne(3,.indent_level,anlrms$_idxrec,.b[bsd$l_vbn],.b[bsd$l_offset]);
3         anl$format_skip(0);
3
3         ! Now the VBN.
3
3         anl$format_line(0,.indent_level+1,anlrms$_idxrecptr,.hp[bkt$v_ptr_sz]+2,
4                         (case .hp[bkt$v_ptr_sz] from 0 to 2 of set
4                         [0]:    .vp[0,0,16,0];
4                         [1]:    .vp[0,0,24,0];
4                         [2]:    .vp[0,0,32,0];
3                         tes));
3
3         ! And the key itself, in hex.
3
3         anl$format_line(0,.indent_level+1,anlrms$_idxkeybytes);
3
4         begin
4         local
4                 key_dsc: descriptor;
4
4         build_descriptor(key_dsc,.key_length,.sp);
4         anl$format_hex(.indent_level+2,key_dsc);
3         end;
2 );
```

```
;   529        P 1027  2 statistics_callback(
;   530        P 1028  2
;   531        P 1029  2          ! If we are accumulating statistics, then we have to call the
;   532        P 1030  2          ! index record callback routine, telling it the level, nominal
;   533        P 1031  2          ! record length, and compressed record length.
;   534        P 1032  2
;   535        P 1033  2          anl$index_callback(.hp[bkt$b_level],
;   536        P 1034  2                             .kp[key$b_keysz] + .hp[bkt$v_ptr_sz]+2,
;   537        P 1035  2                             .key_length + .hp[bkt$v_ptr_sz]+2);
;   538          1036  2 );
;   539          1037  2
;   540          1038  2 ! Now we can advance to the next index record.  If there isn't another
;   541          1039  2 ! one, then just return without modifying the BSD.  Otherwise update the
;   542          1040  2 ! BSD.  Don't forget to increment the record number in the work longword.
;   543          1041  2
;   544          1042  3 if .b[bsd$l_offset]+.key_length lssu .hp[bkt$w_keyfrespc] then (
;   545          1043  3          b[bsd$l_offset] = .b[bsd$l_offset] + .key_length;
;   546          1044  3          increment (b[bsd$l_work]);
;   547          1045  3          return true;
;   548          1046  2 ) else
;   549          1047  2          return false;
;   550          1048  2
;   551          1049  1 end;



                                                    0FFC 00000          .ENTRY   ANL$3INDEX_RECORD, Save R2,R3,R4,R5,R6,R7,-  ; 0959
                                                                                 R8,R9,R10,R11                                 :
                        5B      0000G   CF     9E 00002                 MOVAB    ANL$FORMAT_LINE, R11                          :
                        5E              08     C2 00007                 SUBL2    #8, SP                                        :
                        53         04   AC     D0 0000A                 MOVL     REC_BSD, R3                                   ; 0962
                        50         08   AC     D0 0000E                 MOVL     KEY_BSD, R0                                   ; 0963
                        55         0C   A3     D0 00012                 MOVL     12(R3), HP                                    ; 0977
             5A      0C A3         08   A3     C1 00016                 ADDL3    8(R3), 12(R3), SP                             ; 0978
             57      0C A0         08   A0     C1 0001C                 ADDL3    8(R0), 12(R0), KP                             ; 0979
             08      10 A7              03     E1 00022                 BBC      #3, 16(KP), 1$                                ; 0981
                        56              6A     9A 00027                 MOVZBL   (SP), KEY_LENGTH                              ; 0982
                        56              02     C0 0002A                 ADDL2    #2, KEY_LENGTH                                :
                                        04     11 0002D                 BRB      2$                                            :
                        56         14   A7     9A 0002F 1$:             MOVZBL   20(KP), KEY_LENGTH                            ; 0984
             59         56         08   A3     C1 00033 2$:             ADDL3    8(R3), KEY_LENGTH, R9                         ; 0988
   59     04 A5         10              00     ED 00038                 CMPZV    #0, #16, 4(HP), R9                            :
                                        1B     1E 0003E                 BGEQU    3$                                            :
                                   04   A3     DD 00040                 PUSHL    4(R3)                                         ; 0989
                            00000000G   8F     DD 00043                 PUSHL    #ANLRMS$_BAD3IDXKEYFIT                        :
                 0000G  CF              02     FB 00049                 CALLS    #2, ANL$FORMAT_ERROR                          :
                            00000000G   8F     DD 0004E                 PUSHL    #ANLRMS$_UNWIND                               ; 0990
             00000000G  00              01     FB 00054                 CALLS    #1, LIB$SIGNAL                                :
             50      14 A3              01     C1 0005B 3$:             ADDL3    #1, 20(R3), R0                                ; 0996
   54     0D A5         02              03     EF 00060                 EXTZV    #3, #2, 13(HP), R4                            :
                        58         02   A4     9E 00066                 MOVAB    2(R4), R8                                     :
                        50              58     C4 0006A                 MULL2    R8, R0                                        :
             52      10 A3              50     C3 0006D                 SUBL3    R0, 16(R3), R2                                :
                        52              04     C2 00072                 SUBL2    #4, VP                                        :
                        65         0C   AC     E9 00075                 BLBC     REPORT, 9$                                    ; 0999
```

```
                                7E          04    A3 7D 00079       MOVQ     4(R3), -(SP)                        ; 1003
                                    000000000G    8F DD 0007D       PUSHL    #ANLRMS$_IDXREC                     ;
                                            10    AC DD 00083       PUSHL    INDENT_LEVEL                        ;
                                                  03 DD 00086       PUSHL    #3                                  ;
                                6B                05 FB 00088       CALLS    #5, ANL$FORMAT_LINE                 ;
                                                  7E D4 0008B       CLRL     -(SP)                               ; 1004
                          0000G CF                01 FB 0008D       CALLS    #1, ANL$FORMAT_SKIP                 ;
                    02          00                54 CF 00092       CASEL    R4, #0, #2                          ; 1009
                  0012        0008              0006    00096 4$:   .WORD    5$-4$,-                             ;
                                                                             6$-4$,-                             ;
                                                                             7$-4$                               ;
                                7E                62 3C 0009C 5$:   MOVZWL   (VP), -(SP)                         ; 1010
                                                  09 11 0009F       BRB      8$                                  ;
       7E           62          18                00 EF 000A1 6$:   EXTZV    #0, #24, (VP), -(SP)                ; 1011
                                                  02 11 000A6       BRB      8$                                  ;
                                                  62 DD 000A8 7$:   PUSHL    (VP)                                ; 1012
                                                  58 DD 000AA 8$:   PUSHL    R8                                  ; 1008
                                    000000000G    8F DD 000AC       PUSHL    #ANLRMS$_IDXRECPTR                  ;
                    52       10 AC                01 C1 000B2       ADDL3    #1, INDENT_LEVEL, R2                ;
                                                  52 DD 000B7       PUSHL    R2                                  ;
                                                  7E D4 000B9       CLRL     -(SP)                               ;
                                6B                05 FB 000BB       CALLS    #5, ANL$FORMAT_LINE                 ;
                                    000000000G    8F DD 000BE       PUSHL    #ANLRMS$_IDXKEYBYTES                ; 1017
                                                  52 DD 000C4       PUSHL    R2                                  ;
                                                  7E D4 000C6       CLRL     -(SP)                               ;
                                6B                03 FB 000C8       CALLS    #3, ANL$FORMAT_LINE                 ;
                                6E                56 D0 000CB       MOVL     KEY_LENGTH, KEY_DSC                 ; 1023
                             04 AE                5A D0 000CE       MOVL     SP, KEY_DSC+4                       ;
                                                  5E DD 000D2       PUSHL    SP                                  ; 1024
                    7E       10 AC                02 C1 000D4       ADDL3    #2, INDENT_LEVEL, -(SP)             ;
                          0000G CF                02 FB 000D9       CALLS    #2, ANL$FORMAT_HEX                  ;
                                02       0000G    CF 91 000DE 9$:   CMPB     ANL$GB_MODE, #2                     ; 1036
                                                  07 13 000E3       BEQL     10$                                 ;
                                04       0000G    CF 91 000E5       CMPB     ANL$GB_MODE, #4                     ;
                                                  15 12 000EA       BNEQ     11$                                 ;
                                            02  A446 9F 000EC 10$:  PUSHAB   2(R4)[KEY_LENGTH]                   ;
                                50          14    A7 9A 000F0       MOVZBL   20(KP), R0                          ;
                                            02  A440 9F 000F4       PUSHAB   2(R4)[R0]                           ;
                                7E          0C    A5 9A 000F8       MOVZBL   12(HP), -(SP)                       ;
                          0000G CF                03 FB 000FC       CALLS    #3, ANL$INDEX_CALLBACK              ;
       59        04 A5          10                00 ED 00101 11$:  CMPZV    #0, #16, 4(HP), R9                  ; 1042
                                                  0B 1B 00107       BLEQU    12$                                 ;
                             08 A3                56 C0 00109       ADDL2    KEY_LENGTH, 8(R3)                   ; 1043
                                            14    A3 D6 0010D       INCL     20(R3)                              ; 1044
                                50                01 D0 00110       MOVL     #1, R0                              ; 1047
                                                     04 00113       RET                                          ;
                                                  50 D4 00114 12$:  CLRL     R0                                  ;
                                                     04 00116       RET                                          ; 1049

; Routine Size:  279 bytes,    Routine Base:  $CODE$ + 0311
```

```
1 %sbttl 'ANL$3PRIMARY_DATA_RECORD - Format and Check a Primary Data Record'
1 !++
1 ! Functional Description:
1 !       This routine is responsible for formatting and checking the contents
1 !       of a primary data record for prolog 3 indexed files.  This does not
1 !       include formatting of the data bytes themselves.
1 !
1 ! Formal Parameters:
1 !       rec_bsd         Address of BSD describing data record.  It is updated
1 !                       to describe the next record.
1 !       key_bsd         Address of BSD for key descriptor of this index.
1 !       report          A boolean, true if we are to print a report.
1 !       indent_level    The indentation level for the report.
1 !
1 ! Implicit Inputs:
1 !       global data
1 !
1 ! Implicit Outputs:
1 !       global data
1 !
1 ! Returned Value:
1 !       True if there is another record, false otherwise.
1 !
1 ! Side Effects:
1 !
1 !--
  
1
2 global routine anl$3primary_data_record(rec_bsd,key_bsd,report,indent_level) = begin
2
2 bind
2       b = .rec_bsd: bsd,
2       k = .key_bsd: bsd;
2
2 own
2       data_flags_def: vector[8,long] initial(
2                                       0,
2                                       0,
2                                       0,
2                                       uplit byte (%ascic 'IRC$V_DELETED'),
2                                       uplit byte (%ascic 'IRC$V_RRV'),
2                                       uplit byte (%ascic 'IRC$V_NOPTRSZ'),
2                                       uplit byte (%ascic 'IRC$V_RU_DELETE'),
2                                       uplit byte (%ascic 'IRC$V_RU_UPDATE')
2                                       );
2
2 local
2       hp: ref block[,byte],
2       rp: ref block[,byte],
2       kp: ref block[,byte],
2       overall_dsc: descriptor,
2       key_dsc: descriptor,
2       data_dsc: descriptor;
2
2
2 ! We need to ensure that the data record fits in the used space of the
2 ! bucket.  Begin by making sure that the first byte fits.
```

```
;  610          1107  2
;  611          1108  2 hp = .b[bsd$l_bufptr];
;  612          1109  2
;  613          1110  3 if .b[bsd$l_offset] gequ .hp[bkt$w_freespace] then (
;  614          1111  3         anl$format_error(anlrms$_baddatarecfit,.b[bsd$l_vbn]);
;  615          1112  3         signal (anlrms$_unwind);
;  616          1113  2 );
;  617          1114  2
;  618          1115  2 ! Set up a descriptor of the overall data record, the key, and the data
;  619          1116  2 ! bytes.
;  620          1117  2
;  621          1118  2 calculate_data_record_info(b,k,overall_dsc,key_dsc,data_dsc);
;  622          1119  2
;  623          1120  2 ! Now we can ensure that the entire record fits in the unused space.
;  624          1121  2
;  625          1122  3 if .b[bsd$l_offset]+.overall_dsc[len] gtru .hp[bkt$w_freespace] then (
;  626          1123  3         anl$format_error(anlrms$_baddatarecfit,.b[bsd$l_vbn]);
;  627          1124  3         signal (anlrms$_unwind);
;  628          1125  2 );
```

```
2 ! Now we can format the record, if requested.  This does not include the
2 ! actual data bytes.
2
2 rp = .overall_dsc[ptr];
2 kp = .k[bsd$l_bufptr] + .k[bsd$l_offset];
2
3 if .report then (
3
3         ! Start with a nice heading.
3
3         anl$format_line(3,.indent_level,anlrms$_idxprimrec,.b[bsd$l_vbn],.b[bsd$l_offset]);
3         anl$format_skip(0);
3
3         ! Now the control flags.
3
3         anl$format_flags(.indent_level+1,anlrms$_idxprimrecflags,.rp[irc$b_control],data_flags_def);
3
3         ! Now the record ID.
3
3         anl$format_line(0,.indent_level+1,anlrms$_idxprimrecid,.rp[irc$w_id]);
3
3         ! Now the RRV, both record ID and bucket pointer, if present.
3
3         if not .rp[irc$v_noptrsz] then
3                 anl$format_line(0,.indent_level+1,anlrms$_idxprimrecrrv,
3                                 .rp[irc$w_rrv_id],.rp[irc$v_ptrsz]+2,
4                                 (case .rp[irc$v_ptrsz] from 0 to 2 of set
4                                 [0]:    .rp[5,0,16,0];
4                                 [1]:    .rp[5,0,24,0];
4                                 [2]:    .rp[5,0,32,0];
3                                 tes));
3
3         ! And the key itself, in hex.  It may not exist.
3
4         if not .rp[irc$v_rrv] then (
4                 anl$format_line(0,.indent_level+1,anlrms$_idxkeybytes);
4                 anl$format_hex(.indent_level+2,key_dsc);
3         );
2 );
```

```
  2 ! Now we can actually check the integrity of this data record.  Most of
  2 ! the checking has been done, since it involved the fit of the record
  2 ! in the bucket.  However, we have a few more things to do.
  2
  2 ! Check the control flags.  Don't get confused by the pointer size.
  2
  2 anl$check_flags(.b[bsd$l_vbn],.rp[irc$b_control] and %x'fc',data_flags_def);
  2
  2 ! We don't check the VFC header size since the record might be compressed.
  2
P 2 if not .rp[irc$v_rrv] and not .rp[irc$v_deleted] then statistics_callback(
P 2
P 2         ! If we are accumulating statistics, then we need to call the
P 2         ! statistics callback routine for data records.  It wants the
P 2         ! nominal record length, compressed key length, and compressed
P 2         ! data length.
P 2
P 2         local
P 2                 sp: ref block[,byte],
P 2                 nominal_length: long;
P 2
P 2         ! If the data is compressed, we have to determine its nominal
P 2         ! length by scanning it.  The data record is composed of triplets
P 2         ! of the form (fragment-length,fragment,compression-count).
P 2
P 2         if .kp[key$v_rec_compr] then (
P 2                 sp = .data_dsc[ptr];
P 2                 nominal_length = 0;
P 2
P 2                 while .sp lssa .data_dsc[ptr]+.data_dsc[len] do (
P 2                         nominal_length = .nominal_length + .sp[0,0,16,0];
P 2                         sp = .sp + 2+.sp[0,0,16,0];
P 2                         nominal_length = .nominal_length + .sp[0,0,8,0];
P 2                         increment (sp);
P 2                 );
P 2         );
P 2
P 2         anl$data_callback(.kp[key$b_keysz] +
P 2                                 (if .kp[key$v_rec_compr] then .nominal_length else .data_dsc[len]),
P 2                             .key_dsc[len],
P 2                             .data_dsc[len],
P 2                             0);
  2 );
  2
  2 ! Now we want to advance to the next data record.  If there is room in
  2 ! the bucket for another, then update the BSD.  Otherwise don't touch it.
  2
  3 if .b[bsd$l_offset]+.overall_dsc[len] lssu .hp[bkt$w_freespace] then (
  3         b[bsd$l_offset] = .b[bsd$l_offset] + .overall_dsc[len];
  3         return true;
  2 ) else
  2         return false;
  2
  1 end;
```

```
                                                                                 .PSECT  $PLIT$,NOWRT,NOEXE,2

   44 45 54 45 4C 45 44 5F 56 24 43 52 49 0D  00038 P.AAE:   .ASCII  <13>\IRC$V_DELETED\                       ;
               56 52 52 5F 56 24 43 52 49 09  00046 P.AAF:   .ASCII  <9>\IRC$V_RRV\                            ;
   5A 53 52 54 50 4F 4E 5F 56 24 43 52 49 0D  00050 P.AAG:   .ASCII  <13>\IRC$V_NOPTRSZ\                       ;
54 45 4C 45 44 5F 55 52 5F 56 24 43 52 49 0F  0005E P.AAH:   .ASCII  <15>\IRC$V_RU_DELETE\                     ;
                                          45  0006D                                                            ;
54 41 44 50 55 5F 55 52 5F 56 24 43 52 49 0F  0006E P.AAI:   .ASCII  <15>\IRC$V_RU_UPDATE\                     ;
                                          45  0007D                                                            ;

                                                                                 .PSECT  $OWN$,NOEXE,2

                          00000000  00000000  00000006  0001C DATA_FLAGS_DEF:
                                                                        .LONG    6, 0, 0                       ;
      00000000' 00000000' 00000000' 00000000' 00000000' 00028           .ADDRESS P.AAE, P.AAF, P.AAG, P.AAH, P.AAI ;


                                                                        .PSECT  $CODE$,NOWRT,2

                                                0FFC 00000              .ENTRY   ANL$3PRIMARY_DATA_RECORD, Save R2,R3,R4,R5,-; 1078
                                                                                 R6,R7,R8,R9,R10,R11                  ;
                          5B 00000000G    8F  D0 00002              MOVL     #ANLRMS$_BADDATARECFIT, R11          ;
                          5A     0000G    CF  9E 00009              MOVAB    ANL$FORMAT_LINE, R10                 ;
                          59 00000000G    00  9E 0000E              MOVAB    LIB$SIGNAL, R9                       ;
                          58 00000000G    8F  D0 00015              MOVL     #ANLRMS$_UNWIND, R8                  ;
                          5E              18  C2 0001C              SUBL2    #24, SP                              ;
                          53         04   AC  7D 0001F              MOVQ     REC_BSD, R3                          ; 1081
                          56         0C   A3  D0 00023              MOVL     12(R3), HP                           ; 1108
    08   A3     04   A6   10              00  ED 00027              CMPZV    #0, #16, 4(HP), 8(R3)                ; 1110
                                          0F  1A 0002E              BGTRU    1$                                   ;
                                     04   A3  DD 00030              PUSHL    4(R3)                                ; 1111
                                          5B  DD 00033              PUSHL    R11                                  ;
                    0000G CF              02  FB 00035              CALLS    #2, ANL$FORMAT_ERROR                 ;
                                          58  DD 0003A              PUSHL    R8                                   ; 1112
                          69              01  FB 0003C              CALLS    #1, LIB$SIGNAL                       ;
                                          5E  DD 0003F 1$:          PUSHL    SP                                   ; 1118
                                     0C   AE  9F 00041              PUSHAB   KEY_DSC                              ;
                                     18   AE  9F 00044              PUSHAB   OVERALL_DSC                          ;
                                          18  BB 00047              PUSHR    #^M<R3,R4>                           ;
                    0000V CF              05  FB 00049              CALLS    #5, CALCULATE_DATA_RECORD_INFO       ;
                          57         10   AE  3C 0004E              MOVZWL   OVERALL_DSC, R7                      ; 1122
                          57         08   A3  C0 00052              ADDL2    8(R3), R7                            ;
         57     04   A6   10              00  ED 00056              CMPZV    #0, #16, 4(HP), R7                   ;
                                          0F  1E 0005C              BGEQU    2$                                   ;
                                     04   A3  DD 0005E              PUSHL    4(R3)                                ; 1123
                                          5B  DD 00061              PUSHL    R11                                  ;
                    0000G CF              02  FB 00063              CALLS    #2, ANL$FORMAT_ERROR                 ;
                                          58  DD 00068              PUSHL    R8                                   ; 1124
                          69              01  FB 0006A              CALLS    #1, LIB$SIGNAL                       ;
                          52         14   AE  D0 0006D 2$:          MOVL     OVERALL_DSC+4, RP                    ; 1129
                55     0C A4         08   A4  C1 00071              ADDL3    8(R4), 12(R4), KP                    ; 1130
                          03         0C   AC  E8 00077              BLBS     REPORT, 3$                           ; 1132
                                   009E   31  0007B                 BRW      10$                                  ;
                          7E         04   A3  7D 0007E 3$:          MOVQ     4(R3), -(SP)                         ; 1136
                            00000000G     8F  DD 00082              PUSHL    #ANLRMS$_IDXPRIMREC                  ;
                                     10   AC  DD 00088              PUSHL    INDENT_LEVEL                         ;
```

```
                                             03  DD 0008B       PUSHL    #3                                      :
                              6A             05  FB 0008D       CALLS    #5, ANL$FORMAT_LINE                     :
                                             7E  D4 00090       CLRL     -(SP)                                   : 1137
                        0000G CF             01  FB 00092       CALLS    #1, ANL$FORMAT_SKIP                     :
                                      0000'  CF  9F 00097       PUSHAB   DATA_FLAGS_DEF                          : 1141
                              7E             62  9A 0009B       MOVZBL   (RP), -(SP)                             :
                                  00000000G  8F  DD 0009E       PUSHL    #ANLRMS$_IDXPRIMRECFLAGS                :
                   54      10 AC             01  C1 000A4       ADDL3    #1, INDENT_LEVEL, R4                    :
                                             54  DD 000A9       PUSHL    R4                                      :
                        0000G CF             04  FB 000AB       CALLS    #4, ANL$FORMAT_FLAGS                    :
                              7E         01  A2  3C 000B0       MOVZWL   1(RP), -(SP)                            : 1145
                                  00000000G  8F  DD 000B4       PUSHL    #ANLRMS$_IDXPRIMRECID                   :
                                             54  DD 000BA       PUSHL    R4                                      :
                                             7E  D4 000BC       CLRL     -(SP)                                   :
                              6A             04  FB 000BE       CALLS    #4, ANL$FORMAT_LINE                     :
                   39         62             04  E0 000C1       BBS      #4, (RP), 9$                            : 1149
    50             62         02             00  EF 000C5       EXTZV    #0, #2, (RP), R0                        : 1152
                   02         00             50  CF 000CA       CASEL    R0, #0, #2                              :
                 0014       000C       0006         000CE 4$:   .WORD    5$-4$,-                                 :
                                                                         6$-4$,-                                 :
                                                                         7$-4$                                   :
                              7E         05  A2  3C 000D4 5$:   MOVZWL   5(RP), -(SP)                            : 1153
                                             0B  11 000D8       BRB      8$                                      :
    7E      05     A2         18             00  EF 000DA 6$:   EXTZV    #0, #24, 5(RP), -(SP)                   : 1154
                                             03  11 000E0       BRB      8$                                      :
                                         05  A2  DD 000E2 7$:   PUSHL    5(RP)                                   : 1155
    7E             62         02             00  EF 000E5 8$:   EXTZV    #0, #2, (RP), -(SP)                     : 1151
                              6E             02  C0 000EA       ADDL2    #2, (SP)                                :
                              7E         03  A2  3C 000ED       MOVZWL   3(RP), -(SP)                            :
                                  00000000G  8F  DD 000F1       PUSHL    #ANLRMS$_IDXPRIMRECRRV                  : 1150
                                             54  DD 000F7       PUSHL    R4                                      :
                                             7E  D4 000F9       CLRL     -(SP)                                   :
                              6A             06  FB 000FB       CALLS    #6, ANL$FORMAT_LINE                     :
                   1A         62             03  E0 000FE 9$:   BBS      #3, (RP), 10$                           : 1160
                                  00000000G  8F  DD 00102       PUSHL    #ANLRMS$_IDXKEYBYTES                    : 1161
                                             54  DD 00108       PUSHL    R4                                      :
                                             7E  D4 0010A       CLRL     -(SP)                                   :
                              6A             03  FB 0010C       CALLS    #3, ANL$FORMAT_LINE                     :
                                         08  AE  9F 0010F       PUSHAB   KEY_DSC                                 : 1162
                   7E      10 AC             02  C1 00112       ADDL3    #2, INDENT_LEVEL, -(SP)                 :
                        0000G CF             02  FB 00117       CALLS    #2, ANL$FORMAT_HEX                      :
                                      0000'  CF  9F 0011C 10$:  PUSHAB   DATA_FLAGS_DEF                          : 1171
                              50             62  9A 00120       MOVZBL   (RP), R0                                :
                   7E         50   FFFFFF03  8F  CB 00123       BICL3    #-253, R0, -(SP)                        :
                                         04  A3  DD 0012B       PUSHL    4(R3)                                   :
                        0000G CF             03  FB 0012E       CALLS    #3, ANL$CHECK_FLAGS                     :
                   5B         62             03  E0 00133       BBS      #3, (RP), 15$                           : 1175
                   57         62             02  E0 00137       BBS      #2, (RP), 15$                           :
                              02      0000G  CF  91 0013B       CMPB     ANL$GB_MODE, #2                         : 1207
                                             07  13 00140       BEQL     11$                                     :
                              04      0000G  CF  91 00142       CMPB     ANL$GB_MODE, #4                         :
                                             49  12 00147       BNEQ     15$                                     :
                                         10  A5  95 00149 11$:  TSTB     16(KP)                                  :
                                             25  18 0014C       BGEQ     13$                                     :
                              51         04  AE  D0 0014E       MOVL     DATA_DSC+4, SP                          :
                                             50  D4 00152       CLRL     NOMINAL_LENGTH                          :
                              54             6E  3C 00154       MOVZWL   DATA_DSC, R4                            :
                                   54      04   AE  C0 00157        ADDL2   DATA_DSC+4, R4                          ;
                                   54           51  D1 0015B 12$:   CMPL    SP, R4                                  ;
                                                13  1E 0015E        BGEQU   13$                                     ;
                                   52           61  3C 00160        MOVZWL  (SP), R2                                ;
                                   50           52  C0 00163        ADDL2   R2, NOMINAL_LENGTH                      ;
                                   51      02 A241  9E 00166        MOVAB   2(R2)[SP], SP                           ;
                                   52           B1  9A 0016B        MOVZBL  (SP)+, R2                               ;
                                   50           52  C0 0016E        ADDL2   R2, NOMINAL_LENGTH                      ;
                                                E8  11 00171        BRB     12$                                     ;
                                                7E  D4 00173 13$:   CLRL    -(SP)                                   ;
                                   7E      04   AE  3C 00175        MOVZWL  DATA_DSC, -(SP)                         ;
                                   7E      10   AE  3C 00179        MOVZWL  KEY_DSC, -(SP)                          ;
                                   51      14   A5  9A 0017D        MOVZBL  20(KP), R1                              ;
                                           10   A5  95 00181        TSTB    16(KP)                                  ;
                                                04  19 00184        BLSS    14$                                     ;
                                   50      0C   AE  3C 00186        MOVZWL  DATA_DSC, R0                            ;
                                              6041  9F 0018A 14$:   PUSHAB  (R0)[R1]                                ;
                           0000G   00           04  FB 0018D        CALLS   #4, ANL$DATA_CALLBACK                   ;
           57      04   A6         10           00  ED 00192 15$:   CMPZV   #0, #16, 4(HP), R7                      ; 1212
                                                0C  1B 00198        BLEQU   16$                                     ;
                                   50      10   AE  3C 0019A        MOVZWL  OVERALL_DSC, R0                         ; 1213
                              08   A3           50  C0 0019E        ADDL2   R0, 8(R3)                               ;
                                   50           01  D0 001A2        MOVL    #1, R0                                  ; 1216
                                                    04 001A5        RET                                             ;
                                                50  D4 001A6 16$:   CLRL    R0                                      ;
                                                    04 001A8        RET                                             ; 1218

; Routine Size:  425 bytes,    Routine Base:  $CODE$ + 0428
```

```
1 %sbttl 'ANL$3FORMAT_DATA_BYTES - Format Actual Primary Record Data Bytes'
1 !++
1 ! Functional Description:
1 !       This routine is responsible for formatting the actual data bytes
1 !       in a primary record for prolog 3 indexed files.  Unlike prolog 2,
1 !       this is a separate routine because it's a bit messy.
1 !
1 ! Formal Parameters:
1 !       indent_level    The indentation level for the report.
1 !       rec_bsd         BSD describing COMPLETE primary record.
1 !       key_bsd         BSD for key descriptor for primary index.
1 !
1 ! Implicit Inputs:
1 !       global data
1 !
1 ! Implicit Outputs:
1 !       global data
1 !
1 ! Returned Value:
1 !       None
1 !
1 ! Side Effects:
1 !
1 !--
1
1
2 global routine anl$3format_data_bytes(indent_level,rec_bsd,key_bsd): novalue = begin
2
2 bind
2       b = .rec_bsd: bsd,
2       k = .key_bsd: bsd;
2
2 local
2       rp: ref block[,byte],
2       overall_dsc: descriptor,
2       key_dsc: descriptor,
2       data_dsc: descriptor;
2
2
2 ! Set up a pointer to the record.
2
2 rp = .b[bsd$l_bufptr] + .b[bsd$l_offset];
2
2 ! Set up descriptors for the overall data record, the key, and the data
2 ! bytes.  We only care about the data bytes.
2
2 calculate_data_record_info(b,k,overall_dsc,key_dsc,data_dsc);
2
2 ! If there any data bytes, then format them in hex.  Otherwise tell the user
2 ! there is no data.
2
2 if .data_dsc[len] nequ 0 then
2       anl$format_hex(.indent_level,data_dsc)
2 else
2       signal(anlrms$_nodata);
2
2 return;
```

```
;  782          1276  2
;  783          1277  1 end;



                                                  0000 00000          .ENTRY  ANL$3FORMAT_DATA_BYTES, Save nothing         ; 1245
                            5E                 18  C2 00002           SUBL2   #24, SP                                       ;
                            50           08    AC  D0 00005           MOVL    REC_BSD, R0                                   ; 1248
          51        0C      A0           08    A0  C1 00009           ADDL3   8(R0), 12(R0), RP                             ; 1260
                                               5E  DD 0000F           PUSHL   SP                                            ; 1265
                                         0C    AE  9F 00011           PUSHAB  KEY_DSC                                       ;
                                         18    AE  9F 00014           PUSHAB  OVERALL_DSC                                   ;
                                         0C    AC  DD 00017           PUSHL   KEY_BSD                                       ;
                                               50  DD 0001A           PUSHL   R0                                            ;
                  0000V  CF                    05  FB 0001C           CALLS   #5, CALCULATE_DATA_RECORD_INFO                ;
                                               6E  B5 00021           TSTW    DATA_DSC                                      ; 1270
                                               08  13 00023           BEQL    1$                                            ;
                                               5E  DD 00025           PUSHL   SP                                            ; 1271
                                         04    AC  DD 00027           PUSHL   INDENT_LEVEL                                  ;
                  0000G  CF                    02  FB 0002A           CALLS   #2, ANL$FORMAT_HEX                            ;
                                                   04 0002F           RET                                                   ;
                                  00000000G    8F  DD 00030 1$:       PUSHL   #ANLRMS$_NODATA                               ; 1273
              00000000G  00                    01  FB 00036           CALLS   #1, LIB$SIGNAL                                ;
                                                   04 0003D           RET                                                   ; 1277

; Routine Size:  62 bytes,    Routine Base:  $CODE$ + 05D1
```

```
1 %sbttl 'CALCULATE_DATA_RECORD_INFO'
1 !++
1 ! Description:  This routine is called to calculate the lengths of the various
1 !               portions of a primary data record:  the overall length, the
1 !               key length, and the data bytes length.  This is a complex
1 !               process, particularly with the advent of recovery units.
1 !
1 ! Parameters:   rec_bsd         By reference, the BSD for the data record.
1 !               key_bsd         By reference, the BSD for the key.
1 !               overall_dsc     By reference, a descriptor to be filled in
1 !                               with a description of the overall record.
1 !               key_dsc         By reference, a descriptor to be filled in
1 !                               with a description of the key.
1 !               data_dsc        By reference, a descriptor to be filled in
1 !                               with a description of the data bytes.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:
1 !--
1
1 GLOBAL ROUTINE calculate_data_record_info(rec_bsd: ref bsd,
1                                          key_bsd: ref bsd,
1                                          overall_dsc: ref descriptor,
1                                          key_dsc: ref descriptor,
1                                          data_dsc: ref descriptor)      : novalue
2 = BEGIN
2
2
2 local
2         rp: ref block[,byte],
2         kp: ref block[,byte],
2         sp: ref block[,byte],
2         bits: long;
2
2
2 ! Set up pointers to the primary data record and the key descriptor.
2
2 rp = .rec_bsd[bsd$l_bufptr] + .rec_bsd[bsd$l_offset];
2 kp = .key_bsd[bsd$l_bufptr] + .key_bsd[bsd$l_offset];
2
2 ! The format of a primary data record depends upon the following five things:
2 !       variable-length record
2 !       key compression enabled
2 !       data compression enabled
2 !       data bytes have been deleted
2 !       record update in a recovery unit
2 ! Set up a 5-bit integer specifying the states of these items.
2
2 bits =  ((.anl$gl_fat[fat$v_rtype] nequ fat$c_fixed) ^ 4) +
2         (.kp[key$v_key_compr] ^ 3) +
2         (.kp[key$v_rec_compr] ^ 2) +
2         (.rp[irc$v_deleted]  ^ 1) +
2         .rp[irc$v_ru_update];
2
2 ! Fill in the overall descriptor with the address of the record and the
2 ! length of the overhead portion.
```

```
2
2 overall_dsc[ptr] = .rp;
2 overall_dsc[len] =        1 +
2                           2 +
3                           (if .rp[irc$v_noptrsz] then 0 else
4                                   (case .rp[irc$v_ptrsz] from 0 to 3 of set
4                                   [0]:    4;
4                                   [1]:    5;
4                                   [2]:    6;
5                                   [3]:    (anl$format_error(anlrms$_baddatarecps,.rec_bsd[bsd$l_vbn]);
4                                           signal(anlrms$_unwind););
4                                   tes)
2                           );
2
2 ! Set up a pointer to the portion of the record following the overhead.
2
2 sp = .rp + .overall_dsc[len];
2
2 ! Clear the key and data byte descriptors under the assumption that these
2 ! portions of the record do not exist.
2
2 key_dsc[len] = data_dsc[len] = 0;
2
2 ! If this record is not an RRV, then we need to analyze the key and data
2 ! portions.  Case on the bits we set up to determine the format of these
2 ! portions, and fill in the overall, key, and data byte descriptors.
2
2 if not .rp[irc$v_rrv] then
2         case .bits from 0 to 31 of set
2
2         [%b'00000',
3          %b'00001']:      (overall_dsc[len] = .overall_dsc[len] + .anl$gl_fat[fat$w_maxrec];
3                           key_dsc[len] = .kp[key$b_keysz];
3                           key_dsc[ptr] = .sp;
3                           data_dsc[len] = .anl$gl_fat[fat$w_maxrec] - .key_dsc[len];
2                           data_dsc[ptr] = .sp + .key_dsc[len];);
2
3         [%b'00010']:      (overall_dsc[len] = .overall_dsc[len] + .kp[key$b_keysz];
3                           key_dsc[len] = .kp[key$b_keysz];
2                           key_dsc[ptr] = .sp;);
2
2         [%b'00100',
2          %b'00110',
2          %b'10000',
2          %b'10010',
2          %b'10100',
3          %b'10110']:      (overall_dsc[len] = .overall_dsc[len] + 2+.sp[0,0,16,0];
3                           key_dsc[len] = .kp[key$b_keysz];
3                           key_dsc[ptr] = .sp + 2;
3                           data_dsc[len] = .sp[0,0,16,0] - .key_dsc[len];
2                           data_dsc[ptr] = .sp + 2 + .key_dsc[len];);
2
2         [%b'00101',
2          %b'10001',
3          %b'10101']:      (bind
3                                   real_length = .sp + .sp[0,0,16,0]: word;
3
;  899          1392  3                          overall_dsc[len] = .overall_dsc[len] + 2+.sp[0,0,16,0];
;  900          1393  3                          key_dsc[len] = .kp[key$b_keysz];
;  901          1394  3                          key_dsc[ptr] = .sp + 2;
;  902          1395  3                          data_dsc[len] = .real_length - .key_dsc[len];
;  903          1396  2                          data_dsc[ptr] = .sp + 2 + .key_dsc[len];);
;  904          1397  2
;  905          1398  2         [%b'01000',
;  906          1399  2          %b'01010',
;  907          1400  2          %b'01100',
;  908          1401  2          %b'01110',
;  909          1402  2          %b'11000',
;  910          1403  2          %b'11010',
;  911          1404  2          %b'11100',
;  912          1405  3          %b'11110']:     (overall_dsc[len] = .overall_dsc[len] + 2+.sp[0,0,16,0];
;  913          1406  3                          key_dsc[len] = irc$c_keycmpovh + .sp[2,0,8,0];
;  914          1407  3                          key_dsc[ptr] = .sp + 2;
;  915          1408  3                          data_dsc[len] = .sp[0,0,16,0] - .key_dsc[len];
;  916          1409  2                          data_dsc[ptr] = .sp + 2 + .key_dsc[len];);
;  917          1410  2
;  918          1411  2         [%b'01001',
;  919          1412  2          %b'01101',
;  920          1413  2          %b'11001',
;  921          1414  3          %b'11101']:     (bind
;  922          1415  3                                  real_length = .sp + .sp[0,0,16,0]: word;
;  923          1416  3
;  924          1417  3                          overall_dsc[len] = .overall_dsc[len] + 2+.sp[0,0,16,0];
;  925          1418  3                          key_dsc[len] = irc$c_keycmpovh + .sp[2,0,8,0];
;  926          1419  3                          key_dsc[ptr] = .sp + 2;
;  927          1420  3                          data_dsc[len] = .real_length - .key_dsc[len];
;  928          1421  2                          data_dsc[ptr] = .sp + 2 + .key_dsc[len];);
;  929          1422  2
;  930          1423  2         [inrange,
;  931          1424  3          outrange]:     (anl$format_error(anlrms$_baddatarecbits,.rec_bsd[bsd$l_vbn]);
;  932          1425  2                          signal(anlrms$_unwind););
;  933          1426  2         tes;
;  934          1427  2
;  935          1428  2 ! Ensure that the key and data bytes fit in the overall record.
;  936          1429  2
;  937          1430  2 if .key_dsc[ptr]+.key_dsc[len] gtru .overall_dsc[ptr]+.overall_dsc[len] or
;  938          1431  2    .data_dsc[ptr]+.data_dsc[len] gtru .overall_dsc[ptr]+.overall_dsc[len] then
;  939          1432  2         anl$format_error(anlrms$_badkeydatafit,.rec_bsd[bsd$l_vbn]);
;  940          1433  2
;  941          1434  2 return;
;  942          1435  2
;  943          1436  1 END;
; INFO#212             L1:1345
; Null expression appears in value-required context



                                    0FFC 00000          .ENTRY  CALCULATE_DATA_RECORD_INFO, Save R2,R3,R4,-  ; 1299
                                                                R5,R6,R7,R8,R9,R10,R11                       ;
              5B 00000000G   00   9E 00002          MOVAB   LIB$SIGNAL, R11                              ;
              5A 00000000G   8F   D0 00009          MOVL    #ANLRMS$_UNWIND, R10                         ;
              57        04   AC   D0 00010          MOVL    REC_BSD, R7                                  ; 1316
```

```
                           58        0C  A7        08   A7 C1 00014         ADDL3   8(R7), 12(R7), RP                       ;
                                         50        08   AC D0 0001A         MOVL    KEY_BSD, R0                             ; 1317
                           56        0C  A0        08   A0 C1 0001E         ADDL3   8(R0), 12(R0), KP                       ;
                                                        51 D4 00024         CLRL    R1                                      ; 1327
            01     0000G   DF            04             00 ED 00026         CMPZV   #0, #4, @ANL$GL_FAT, #1                 ;
                                                        02 13 0002D         BEQL    1$                                      ;
                                                        51 D6 0002F         INCL    R1                                      ;
                                         51             10 C4 00031 1$:     MULL2   #16, R1                                 ;
            50        10   A6            01             06 EF 00034         EXTZV   #6, #1, 16(KP), R0                      ; 1328
                                         51           6140 7E 0003A         MOVAQ   (R1)[R0], R1                            ; 1327
            50        10   A6            01             07 EF 0003E         EXTZV   #7, #1, 16(KP), R0                      ; 1329
                                         51           6140 DE 00044         MOVAL   (R1)[R0], R1                            ; 1328
            50             68            01             02 EF 00048         EXTZV   #2, #1, (RP), R0                        ; 1330
                                         50           6140 3E 0004D         MOVAW   (R1)[R0], R0                            ; 1329
            59             68            01             06 EF 00051         EXTZV   #6, #1, (RP), BITS                      ; 1331
                                         59             50 C0 00056         ADDL2   R0, BITS                                ;
                                         55        CC   AC D0 00059         MOVL    OVERALL_DSC, R5                         ; 1336
                                     04  A5             58 D0 0005D         MOVL    RP, 4(R5)                               ;
                           33            68             04 E0 00061         BBS     #4, (RP), 7$                            ; 1339
            52             68            02             00 EF 00065         EXTZV   #0, #2, (RP), R2                        ; 1340
                           03            00             52 CF 0006A         CASEL   R2, #0, #3                              ;
          0017           0012          000D           0008    0006E 2$:     .WORD   3$-2$,-                                 ;
                                                                                    4$-2$,-                                 ;
                                                                                    5$-2$,-                                 ;
                                                                                    6$-2$                                   ;
                                         50             04 D0 00076 3$:     MOVL    #4, R0                                  ;
                                                        1F 11 00079         BRB     8$                                      ;
                                         50             05 D0 0007B 4$:     MOVL    #5, R0                                  ;
                                                        1A 11 0007E         BRB     8$                                      ;
                                         50             06 D0 00080 5$:     MOVL    #6, R0                                  ;
                                                        15 11 00083         BRB     8$                                      ;
                                                   04   A7 DD 00085 6$:     PUSHL   4(R7)                                   ; 1344
                                           000000000G   8F DD 00088         PUSHL   #ANLRMS$_BADDATARECPS                   ;
                                  0000G  CF             02 FB 0008E         CALLS   #2, ANL$FORMAT_ERROR                    ;
                                                        5A DD 00093         PUSHL   R10                                     ; 1345
                                         68             01 FB 00095         CALLS   #1, LIB$SIGNAL                          ;
                                                        50 D4 00098 7$:     CLRL    R0                                      ; 1340
                           65            50             03 A1 0009A 8$:     ADDW3   #3, R0, (R5)                            ; 1338
                                         54             65 3C 0009E         MOVZWL  (R5), SP                                ; 1351
                                         54             58 C0 000A1         ADDL2   RP, SP                                  ;
                                         53        10   AC D0 000A4         MOVL    KEY_DSC, R3                             ; 1356
                                         52        14   AC D0 000A8         MOVL    DATA_DSC, R2                            ;
                                                        62 B4 000AC         CLRW    (R2)                                    ;
                                                        63 B4 000AE         CLRW    (R3)                                    ;
                           77            68             03 E0 000B0         BBS     #3, (RP), 12$                           ; 1362
                           1F            00             59 CF 000B4         CASEL   BITS, #0, #31                           ; 1363
          0040           0075          0055           0055    000B8 9$:     .WORD   11$-9$,-                                ;
          0040           0086          0099           0086    000C0                 11$-9$,-                                ;
          0040           00AD          00CF           00AD    000C8                 13$-9$,-                                ;
          0040           00AD          00CF           00AD    000D0                 10$-9$,-                                ;
          0040           0086          0099           0086    000D8                 15$-9$,-                                ;
          0040           0086          0099           0086    000E0                 16$-9$,-                                ;
          0040           00AD          00CF           00AD    000E8                 15$-9$,-                                ;
          0040           00AD          00CF           00AD    000F0                 10$-9$,-                                ;
                                                                                    17$-9$,-                                ;
                                                                                    19$-9$,-                                ;
                                                                                    17$-9$,-                                ;
```

```
                                                                   10$-9$,-                              ;
                                                                   17$-9$,-                              ;
                                                                   19$-9$,-                              ;
                                                                   17$-9$,-                              ;
                                                                   10$-9$,-                              ;
                                                                   15$-9$,-                              ;
                                                                   16$-9$,-                              ;
                                                                   15$-9$,-                              ;
                                                                   10$-9$,-                              ;
                                                                   15$-9$,-                              ;
                                                                   16$-9$,-                              ;
                                                                   15$-9$,-                              ;
                                                                   10$-9$,-                              ;
                                                                   17$-9$,-                              ;
                                                                   19$-9$,-                              ;
                                                                   17$-9$,-                              ;
                                                                   10$-9$,-                              ;
                                                                   17$-9$,-                              ;
                                                                   19$-9$,-                              ;
                                                                   17$-9$,-                              ;
                                                                   10$-9$                                ;
                             04   A7   DD 000F8 10$:       PUSHL   4(R7)                                 ; 1424
                      00000000G   8F   DD 000FB            PUSHL   #ANLRMS$_BADDATAREC BITS              ;
             0000G  CF            02   FB 00101            CALLS   #2, ANL$FORMAT_ERROR                  ;
                                  5A   DD 00106            PUSHL   R10                                   ; 1425
                    6B            01   FB 00108            CALLS   #1, LIB$SIGNAL                        ;
                                  2F   11 0010B            BRB     14$                                   ; 1363
                    50     0000G  CF   D0 0010D 11$:       MOVL    ANL$GL_FAT, R0                        ; 1366
                    65        10  A0   A0 00112            ADDW2   16(R0), (R5)                          ;
                    63        14  A6   9B 00116            MOVZBW  20(KP), (R3)                          ; 1367
               04   A3            54   D0 0011A            MOVL    SP, 4(R3)                             ; 1368
         62    10   A0            63   A3 0011E            SUBW3   (R3), 16(R0), (R2)                    ; 1369
                    50            63   3C 00123            MOVZWL  (R3), R0                              ; 1370
     04  A2         54            50   C1 00126            ADDL3   R0, SP, 4(R2)                         ;
                                  0F   11 0012B 12$:       BRB     14$                                   ; 1363
                    50        14  A6   9A 0012D 13$:       MOVZBL  20(KP), R0                            ; 1372
                    65            50   A0 00131            ADDW2   R0, (R5)                              ;
                    63        14  A6   9B 00134            MOVZBW  20(KP), (R3)                          ; 1373
               04   A3            54   D0 00138            MOVL    SP, 4(R3)                             ; 1374
                                  73   11 0013C 14$:       BRB     22$                                   ; 1363
                    50            65   3C 0013E 15$:       MOVZWL  (R5), R0                              ; 1381
                    51            64   3C 00141            MOVZWL  (SP), R1                              ;
                    50            51   C0 00144            ADDL2   R1, R0                                ;
         65         50            02   A1 00147            ADDW3   #2, R0, (R5)                          ;
                    63        14  A6   9B 0014B            MOVZBW  20(KP), (R3)                          ; 1382
                                  28   11 0014F            BRB     18$                                   ; 1383
                    51            64   3C 00151 16$:       MOVZWL  (SP), R1                              ; 1390
                    50            65   3C 00154            MOVZWL  (R5), R0                              ; 1392
                    58       02 A140   9E 00157            MOVAB   2(R1)[R0], R8                         ;
                    65            58   B0 0015C            MOVW    R8, (R5)                              ;
                    63        14  A6   9B 0015F            MOVZBW  20(KP), (R3)                          ; 1393
                                  37   11 00163            BRB     20$                                   ; 1394
                    50            65   3C 00165 17$:       MOVZWL  (R5), R0                              ; 1405
                    51            64   3C 00168            MOVZWL  (SP), R1                              ;
                    50            51   C0 0016B            ADDL2   R1, R0                                ;
         65         50            02   A1 0016E            ADDW3   #2, R0, (R5)                          ;
                    63        02  A4   9B 00172            MOVZBW  2(SP), (R3)                           ; 1406
```

```
                      63           02   A0 00176         ADDW2    #2, (R3)                                   ;
               04     A3      02   A4   9E 00179 18$:    MOVAB    2(SP), 4(R3)                               ; 1407
                      50           63   3C 0017E         MOVZWL   (R3), R0                                   ; 1408
         62           64           50   A3 00181         SUBW3    R0, (SP), (R2)                             ;
                                   24   11 00185         BRB      21$                                        ; 1409
                      51           64   3C 00187 19$:    MOVZWL   (SP), R1                                   ; 1415
                      50           65   3C 0018A         MOVZWL   (R5), R0                                   ; 1417
                      56      02 A140   9E 0018D         MOVAB    2(R1)[R0], R6                              ;
                      65           56   B0 00192         MOVW     R6, (R5)                                   ;
                      63      02   A4   9B 00195         MOVZBW   2(SP), (R3)                                ; 1418
                      63           02   A0 00199         ADDW2    #2, (R3)                                   ;
               04     A3      02   A4   9E 0019C 20$:    MOVAB    2(SP), 4(R3)                               ; 1419
                      50           63   3C 001A1         MOVZWL   (R3), R0                                   ; 1420
                                 6144   9F 001A4         PUSHAB   (R1)[SP]                                   ;
         62           9E           50   A3 001A7         SUBW3    R0, @(SP)+, (R2)                           ;
               04     A2      02 A044   9E 001AB 21$:    MOVAB    2(R0)[SP], 4(R2)                           ; 1421
                      50           63   3C 001B1 22$:    MOVZWL   (R3), R0                                   ; 1430
         53           50      04   A3   C1 001B4         ADDL3    4(R3), R0, R3                              ;
                      50           65   3C 001B9         MOVZWL   (R5), R0                                   ;
         55           50      04   A5   C1 001BC         ADDL3    4(R5), R0, R5                              ;
                      55           53   D1 001C1         CMPL     R3, R5                                     ;
                                   0D   1A 001C4         BGTRU    23$                                        ;
                      50           62   3C 001C6         MOVZWL   (R2), R0                                   ; 1431
         52           50      04   A2   C1 001C9         ADDL3    4(R2), R0, R2                              ;
                      55           52   D1 001CE         CMPL     R2, R5                                     ;
                                   0E   1B 001D1         BLEQU    24$                                        ;
                              04   A7   DD 001D3 23$:    PUSHL    4(R7)                                      ; 1432
                         00000000G 8F   DD 001D6         PUSHL    #ANLRMS$_BADKEYDATAFIT                     ;
             0000G    CF           02   FB 001DC         CALLS    #2, ANL$FORMAT_ERROR                       ;
                                        04 001E1 24$:    RET                                                 ; 1436

; Routine Size:  482 bytes,    Routine Base:  $CODE$ + 060F
```

```
1 %sbttl 'ANL$3SIDR_RECORD - Print & Check a Secondary Data Record'
1 !++
1 ! Functional Description:
1 !       This routine is responsible for printing and checking the contents
1 !       of a prologue 3 secondary data record (SIDR).  SIDRs exist in the
1 !       data buckets of secondary indices.
1 !
1 ! Formal Parameters:
1 !       rec_bsd         Address of BSD describing the SIDR.
1 !                       The BSD is updated to describe the next SIDR.
1 !       key_bsd         Address of BSD describing the key for this index.
1 !       report          A boolean, true if we are to format the SIDR.
1 !       indent_level    Indentation level for the report, if formatted.
1 !
1 ! Implicit Inputs:
1 !       global data
1 !
1 ! Implicit Outputs:
1 !       global data
1 !
1 ! Returned Value:
1 !       True if there is another SIDR in the bucket, false if not.
1 !
1 ! Side Effects:
1 !
1 !--
1
1
1 global routine anl$3sidr_record(rec_bsd,
1                                 key_bsd,
1                                 report: byte,
2                                 indent_level: long)     = begin
2
2 bind
2       b = .rec_bsd: bsd,
2       k = .key_bsd: bsd;
2
2 local
2       hp: ref block[,byte],
2       sp: ref block[,byte],
2       kp: ref block[,byte],
2       length: long,
2       key_length: long,
2       p: bsd,
2       sidr_pointers: long;
2
2
2 ! First we have to ensure that the SIDR record fits in the used space of
2 ! the bucket.  If not, we have a drastic structure error.  Begin by ensuring
2 ! that the length, which is the first word, fits.
2
2 hp = .b[bsd$l_bufptr];
3 if .b[bsd$l_offset] + 1 gequ .hp[bkt$w_freespace] then (
3       anl$format_error(anlrms$_baddatarecfit,.b[bsd$l_vbn]);
3       signal (anlrms$_unwind);
2 );
2
```

```
2 ! Now we calculate the length of the entire SIDR record.  It's just the
2 ! 2-byte length plus the number of bytes specified by the length.  While
2 ! we're at it, calculate the length of the key.
2
2 kp = .k[bsd$l_bufptr] + .k[bsd$l_offset];
2 sp = .b[bsd$l_bufptr] + .b[bsd$l_offset];
2 length =        2 +
2                 .sp[0,0,16,0];
3 key_length =    (if .kp[key$v_key_compr] then
3                         .sp[2,0,8,0] + irc$c_keycmpovh
3                 else
2                         .kp[key$b_keysz]);
2
2 ! Make sure the entire SIDR fits in the used space of the bucket.
2
3 if .b[bsd$l_offset] + .length gtru .hp[bkt$w_freespace] then (
3                 anl$format_error(anlrms$_baddatarecfit,.b[bsd$l_vbn]);
3                 signal (anlrms$_unwind);
2 );
```

```
; 1022          1513  2 ! Now we can format the SIDR record fixed portion, if requested.
; 1023          1514  2
; 1024          1515  3 if .report then (
; 1025          1516  3
; 1026          1517  3         ! Start with a nice header.
; 1027          1518  3
; 1028          1519  3         anl$format_line(3,.indent_level,anlrms$_idxsidr,.b[bsd$l_vbn],.b[bsd$l_offset]);
; 1029          1520  3         anl$format_skip(0);
; 1030          1521  3
; 1031          1522  3         ! All we have to format is the key.  Build a descriptor for it and
; 1032          1523  3         ! dump it in hex.
; 1033          1524  3
; 1034          1525  3         anl$format_line(0,.indent_level+1,anlrms$_idxkeybytes);
; 1035          1526  4         begin
; 1036          1527  4         local
; 1037          1528  4                 key_dsc: descriptor;
; 1038          1529  4
; 1039          1530  4         build_descriptor(key_dsc, .key_length,sp[2,0,0,0]);
; 1040          1531  4         anl$format_hex(.indent_level+2,key_dsc);
; 1041          1532  3         end;
; 1042          1533  2 );
```

```
  2 ! There is nothing more to check about the fixed portion of the SIDR.
  2 ! If we aren't displaying this record, then we want to check all of
  2 ! the SIDR pointers.
  2
  2 sidr_pointers = 0;
  3 if not .report then (
  3
  3         ! Set up a BSD to describe the first SIDR pointer.  This includes
  3         ! setting the work longword to the number of bytes worth of
  3         ! pointer existing in the record.
  3
  3         init_bsd(p);
  3         copy_bucket(b,p);
  3         p[bsd$l_offset] = .b[bsd$l_offset] + 2 + .key_length;
  3         p[bsd$l_work] = .sp[0,0,16,0] - .key_length;
  3
  3         ! Now we can loop through each pointer, checking its integrity,
  3         ! and counting them as we go.
  3
  3         do increment(sidr_pointers) while anl$3sidr_pointer(p,false);
  3
  3         anl$bucket(p,-1);
  2 );
  2
P 2 statistics_callback(
P 2
P 2         ! If we are accumulating statistics, we want to call the data
P 2         ! record callback routine and tell it the overall record length,
P 2         ! compressed key length, and compressed data length.  The latter
P 2         ! makes no sense for SIDRs.  We also need to tell it the number
P 2         ! of SIDR pointers in this record.
P 2
P 2         anl$data_callback(.length,
P 2                           .key_length,
P 2                           0,
P 2                           .sidr_pointers);
  2 );
```

```
; 1082           1571  2 ! Now we want to advance on to the next SIDR in this bucket.  if there
; 1083           1572  2 ! isn't room for one, then we're done.  Otherwise update the BSD.
; 1084           1573  2
; 1085           1574  3 if .b[bsd$l_offset] + .length lssu .hp[bkt$w_freespace] then (
; 1086           1575  3         b[bsd$l_offset] = .b[bsd$l_offset] + .length;
; 1087           1576  3         return true;
; 1088           1577  2 ) else
; 1089           1578  2         return false;
; 1090           1579  2
; 1091           1580  1 end;



                                                OFFC 00000        .ENTRY   ANL$3SIDR_RECORD, Save R2,R3,R4,R5,R6,R7,-    ; 1465
                                                                           R8,R9,R10,R11
                                5E               28  C2 00002        SUBL2    #40, SP
                                56          04   AC  D0 00005        MOVL     REC_BSD, R6                                   ; 1471
                                52          08   AC  D0 00009        MOVL     KEY_BSD, R2                                   ; 1472
                                5A          0C   A6  D0 0000D        MOVL     12(R6), HP                                    ; 1488
                                57          08   A6  D0 00011        MOVL     8(R6), R7                                     ; 1489
                                50          01   A7  9E 00015        MOVAB    1(R7), R0
        50      04   AA         10               00  ED 00019        CMPZV    #0, #16, 4(HP), R0
                                                 1B  1A 0001F        BGTRU    1$
                                            04   A6  DD 00021        PUSHL    4(R6)                                         ; 1490
                                    000000000G   8F  DD 00024        PUSHL    #ANLRMS$_BADDATARECFIT
                       0000G  CF                 02  FB 0002A        CALLS    #2, ANL$FORMAT_ERROR
                                    000000000G   8F  DD 0002F        PUSHL    #ANLRMS$_UNWIND                               ; 1491
                  000000000G  00                 01  FB 00035        CALLS    #1, LIB$SIGNAL
                    50    0C  A2            08   A2  C1 0003C 1$:    ADDL3    8(R2), 12(R2), KP                             ; 1498
                    59        57            0C   A6  C1 00042        ADDL3    12(R6), R7, SP                                ; 1499
                              6E                 69  3C 00047        MOVZWL   (SP), LENGTH                                  ; 1500
                              6E                 02  C0 0004A        ADDL2    #2, LENGTH
                    09    10  A0                 06  E1 0004D        BBC      #6, 16(KP), 2$                                ; 1502
                              58            02   A9  9A 00052        MOVZBL   2(SP), KEY_LENGTH                             ; 1503
                              58                 02  C0 00056        ADDL2    #2, KEY_LENGTH
                                                 04  11 00059        BRB      3$
                              58            14   A0  9A 0005B 2$:    MOVZBL   20(KP), KEY_LENGTH                            ; 1505
               04   AE        57                 6E  C1 0005F 3$:    ADDL3    LENGTH, R7, 4(SP)                             ; 1509
  04   AE      04   AA        10                 00  ED 00064        CMPZV    #0, #16, 4(HP), 4(SP)
                                                 1B  1E 0006B        BGEQU    4$
                                            04   A6  DD 0006D        PUSHL    4(R6)                                         ; 1510
                                    000000000G   8F  DD 00070        PUSHL    #ANLRMS$_BADDATARECFIT
                       0000G  CF                 02  FB 00076        CALLS    #2, ANL$FORMAT_ERROR
                                    000000000G   8F  DD 0007B        PUSHL    #ANLRMS$_UNWIND                               ; 1511
                  000000000G  00                 01  FB 00081        CALLS    #1, LIB$SIGNAL
                              44            0C   AC  E9 00088 4$:    BLBC     REPORT, 5$                                    ; 1515
                                                 57  DD 0008C        PUSHL    R7                                            ; 1519
                                            04   A6  DD 0008E        PUSHL    4(R6)
                                    000000000G   8F  DD 00091        PUSHL    #ANLRMS$_IDXSIDR
                                            10   AC  DD 00097        PUSHL    INDENT_LEVEL
                                                 03  DD 0009A        PUSHL    #3
                       0000G  CF                 05  FB 0009C        CALLS    #5, ANL$FORMAT_LINE
                                                 7E  D4 000A1        CLRL     -(SP)                                         ; 1520
                       0000G  CF                 01  FB 000A3        CALLS    #1, ANL$FORMAT_SKIP
                                    000000000G   8F  DD 000A8        PUSHL    #ANLRMS$_IDXKEYBYTES                          ; 1525
```

```
                          7E        10  AC            01  C1 000AE          ADDL3   #1, INDENT_LEVEL, -(SP)                      ;
                                                      7E  D4 000B3          CLRL    -(SP)                                        ;
                                 0000G  CF            03  FB 000B5          CALLS   #3, ANL$FORMAT_LINE                          ;
                                    08  AE            58  D0 000BA          MOVL    KEY_LENGTH, KEY_DSC                          ; 1530
                                    0C  AE       02   A9  9E 000BE          MOVAB   2(R9), KEY_DSC+4                             ;
                                                 08   AE  9F 000C3          PUSHAB  KEY_DSC                                      ; 1531
                          7E        10  AC            02  C1 000C6          ADDL3   #2, INDENT_LEVEL, -(SP)                      ;
                                 0000G  CF            02  FB 000CB          CALLS   #2, ANL$FORMAT_HEX                           ;
                                                      5B  D4 000D0  5$:     CLRL    SIDR_POINTERS                                ; 1538
                                        47       0C   AC  E8 000D2          BLBS    REPORT, 7$                                   ; 1539
            18            00            6E            00  2C 000D6          MOVC5   #0, (SP), #0, #24, P                         ; 1545
                                                 10   AE     000DB                                                               ;
                                    10  AE            65  7D 000DD          MOVQ    (R6), T                                      ; 1546
                                    18  AE       08   A6  D0 000E1          MOVL    8(R6), T+8                                   ;
                                    24  AE       14   A6  D0 000E6          MOVL    20(R6), T+20                                 ;
                                                      7E  D4 000EB          CLRL    -(SP)                                        ;
                                                 14   AE  9F 000ED          PUSHAB  T                                            ;
                                 0000G  CF            02  FB 000F0          CALLS   #2, ANL$BUCKET                               ;
                                    18  AE       02 A847  9E 000F5          MOVAB   2(KEY_LENGTH)[R7], P+8                       ; 1547
                                        50            69  3C 000FB          MOVZWL  (SP), R0                                     ; 1548
                     24   AE            50            58  C3 000FE          SUBL3   KEY_LENGTH, R0, P+20                         ;
                                                      5B  D6 00103  6$:     INCL    SIDR_POINTERS                                ; 1553
                                                      7E  D4 00105          CLRL    -(SP)                                        ;
                                                 14   AE  9F 00107          PUSHAB  P                                            ;
                                 0000V  CF            02  FB 0010A          CALLS   #2, ANL$3SIDR_POINTER                        ;
                                        F1            50  E8 0010F          BLBS    R0, 6$                                       ;
                                        7E            01  CE 00112          MNEGL   #1, -(SP)                                    ; 1555
                                                 14   AE  9F 00115          PUSHAB  P                                            ;
                                 0000G  CF            02  FB 00118          CALLS   #2, ANL$BUCKET                               ;
                                        02    0000G   CF  91 0011D  7$:     CMPB    ANL$GB_MODE, #2                              ; 1570
                                                      07  13 00122          BEQL    8$                                           ;
                                        04    0000G   CF  91 00124          CMPB    ANL$GB_MODE, #4                              ;
                                                      0E  12 00129          BNEQ    9$                                           ;
                                                      5B  DD 0012B  8$:     PUSHL   SIDR_POINTERS                                ;
                                                      7E  D4 0012D          CLRL    -(SP)                                        ;
                                                      58  DD 0012F          PUSHL   KEY_LENGTH                                   ;
                                                 0C   AE  DD 00131          PUSHL   LENGTH                                       ;
                                 0000G  CF            04  FB 00134          CALLS   #4, ANL$DATA_CALLBACK                        ;
       04   AE       04   AA            10            00  ED 00139  9$:     CMPZV   #0, #16, 4(HP), 4(SP)                        ; 1574
                                                      08  1B 00140          BLEQU   10$                                          ;
                                    08  A6            6E  C0 00142          ADDL2   LENGTH, 8(R6)                                ; 1575
                                        50            01  D0 00146          MOVL    #1, R0                                       ; 1578
                                                          04 00149          RET                                                  ;
                                                      50  D4 0014A  10$:    CLRL    R0                                           ;
                                                          04 0014C          RET                                                  ; 1580

; Routine Size:  333 bytes,    Routine Base:  $CODE$ + 07F1
```

```
1 %sbttl 'ANL$3SIDR_POINTER - Format & Analyze SIDR Pointer'
1 !++
1 ! Functional Description:
1 !       This routine is responsible for formatting and analyzing one of the
1 !       pointers in a SIDR record.  There is one pointer for each record
1 !       having the secondary key present in the SIDR header.  This code is
1 !       for prologue 3 indexed files.
1 !
1 ! Formal Parameters:
1 !       pointer_bsd     Address of BSD describing the pointer.  The work
1 !                       longword in the BSD is assumed to contain a count
1 !                       of remaining bytes in the SIDR record.
1 !       report          Boolean, true if we are to format the pointer.
1 !       indent_level    Indentation level for the report.
1 !
1 ! Implicit Inputs:
1 !       global data
1 !
1 ! Implicit Outputs:
1 !       global data
1 !
1 ! Returned Value:
1 !       True if there is another SIDR pointer, false otherwise.
1 !
1 ! Side Effects:
1 !
1 !--
1
1
1 global routine anl$3sidr_pointer(pointer_bsd,
1                                  report: byte,
2                                  indent_level: long)    = begin
2
2 bind
2       p = .pointer_bsd: bsd;
2
2 own
2       pointer_flags_def: vector[9,long] initial(
2                                 7,
2                                 0,
2                                 0,
2                                 uplit byte (%ascic 'IRC$V_DELETED'),
2                                 0,
2                                 uplit byte (%ascic 'IRC$V_NOPTRSZ'),
2                                 uplit byte (%ascic 'IRC$V_RU_DELETE'),
2                                 0,
2                                 uplit byte (%ascic 'IRC$V_FIRST_KEY')
2                                 );
2
2 local
2       pp: ref block[,byte],
2       length: long;
2
2 ! We know the SIDR record fits in the used space of the bucket, because
2 ! that was checked in ANL$3SIDR_RECORD.
2
```

```
; 1150          1638  2 ! So we can calculate the overall length of the pointer.
; 1151          1639  2
; 1152          1640  2 pp = .p[bsd$l_bufptr] + .p[bsd$l_offset];
; 1153          1641  2 length =        1 +
; 1154          1642  3                 (if .pp[irc$v_noptrsz] then 0 else
; 1155          1643  4                         (case .pp[irc$v_ptrsz] from 0 to 3 of set
; 1156          1644  4                         [0]:    4;
; 1157          1645  4                         [1]:    5;
; 1158          1646  4                         [2]:    6;
; 1159          1647  5                         [3]:    (anl$format_error(anlrms$_baddatarecps,.p[bsd$l_vbn]);
; 1160          1648  4                                 signal (anlrms$_unwind););
; 1161          1649  4                         tes)
; 1162          1650  2                 );
; 1163          1651  2
; 1164          1652  2 ! Make sure the entire pointer fits in the SIDR record.  If not, that's a
; 1165          1653  2 ! drastic structure error.
; 1166          1654  2
; 1167          1655  3 if .length gtru .p[bsd$l_work] then (
; 1168          1656  3         anl$format_error(anlrms$_badsidrptrfit,.p[bsd$l_vbn]);
; 1169          1657  3         signal (anlrms$_unwind);
; 1170          1658  2 );
```

```
2 ! Now we can format the SIDR pointer if requested.
2
3 if .report then (
3
3               ! Format the flags.
3
3               anl$format_flags(.indent_level,anlrms$_idxsidrptrflags,.pp[irc$b_control],pointer_flags_def);
3
3               ! And the record ID and bucket VBN, if present.
3
4               if not .pp[irc$v_noptrsz] then (
4                       anl$format_line(0,.indent_level,anlrms$_idxsidrptrref,.pp[1,0,16,0],.pp[irc$v_ptrsz]+2,
5                                       (case .pp[irc$v_ptrsz] from 0 to 2 of set
5                                       [0]:    .pp[3,0,16,0];
5                                       [1]:    .pp[3,0,24,0];
5                                       [2]:    .pp[3,0,32,0];
4                                       tes));
3               );
2 );
```

```
; 1192          1678  2 ! Now we have to check the record pointer.  The only thing to check is
; 1193          1679  2 ! the control flags.  Don't get confused by the pointer size.
; 1194          1680  2
; 1195          1681  2 anl$check_flags(.p[bsd$l_vbn],.pp[irc$b_control] and %x'fc',pointer_flags_def);
; 1196          1682  2
; 1197          1683  2 ! Now we want to advance on to the next pointer.  Reduce the count of
; 1198          1684  2 ! remaining bytes.  If it goes to zero, there are no more pointers.
; 1199          1685  2 ! If it doesn't, then update the BSD.
; 1200          1686  2
; 1201          1687  2 p[bsd$l_work] = .p[bsd$l_work] - .length;
; 1202          1688  3 if .p[bsd$l_work] gtru 0 then (
; 1203          1689  3         p[bsd$l_offset] = .p[bsd$l_offset] + .length;
; 1204          1690  3         return true;
; 1205          1691  2 ) else
; 1206          1692  2         return false;
; 1207          1693  2
; 1208          1694  1 end;
; INFO#212                  L1:1648
; Null expression appears in value-required context



                                                                         .PSECT  $PLIT$,NOWRT,NOEXE,2

    44  45  54  45  4C  45  44  5F  56  24  43  52  49  0D  0007E P.AAJ:  .ASCII  <13>\IRC$V_DELETED\                       ;
    5A  53  52  54  50  4F  4E  5F  56  24  43  52  49  0D  0008C P.AAK:  .ASCII  <13>\IRC$V_NOPTRSZ\                       ;
54  45  4C  45  44  5F  55  52  5F  56  24  43  52  49  0F  0009A P.AAL:  .ASCII  <15>\IRC$V_RU_DELETE\                     ;
                                                        45  000A9                                                          ;
45  4B  5F  54  53  52  49  46  5F  56  24  43  52  49  0F  000AA P.AAM:  .ASCII  <15>\IRC$V_FIRST_KEY\                     ;
                                                        59  000B9                                                          ;

                                                                         .PSECT  $OWN$,NOEXE,2

                             00000000  00000000  00000007  0003C POINTER_FLAGS_DEF:
                                                                         .LONG   7, 0, 0                                   ;
                                                 00000000' 00048         .ADDRESS P.AAJ                                    ;
                                                 00000000  0004C         .LONG   0                                         ;
                                       00000000' 00000000' 00050         .ADDRESS P.AAK, P.AAL                             ;
                                                 00000000  00058         .LONG   0                                         ;
                                                 00000000' 0005C         .ADDRESS P.AAM                                    ;


                                                                         .PSECT  $CODE$,NOWRT,2

                                                      00FC 00000         .ENTRY  ANL$3SIDR_POINTER, Save R2,R3,R4,R5,R6,R7 ; 1610
                                       57 00000000G  00  9E 00002         MOVAB   LIB$SIGNAL, R7                            ;
                                       56 00000000G  8F  D0 00009         MOVL    #ANLRMS$_UNWIND, R6                       ;
                                       54        04  AC  D0 00010         MOVL    POINTER_BSD, R4                           ; 1615
                         52        0C  A4        08  A4  C1 00014         ADDL3   8(R4), 12(R4), PP                         ; 1640
                         33            62            04  E0 0001A         BBS     #4, (PP), 6$                              ; 1642
           55            62            02            00  EF 0001E         EXTZV   #0, #2, (PP), R5                          ; 1643
                         03            00            55  CF 00023         CASEL   R5, #0, #3                                ;
         0017          0012          000D         0008      00027 1$:     .WORD   2$-1$,-                                   ;
                                                                                  3$-1$,-                                   ;
                                                                                  4$-1$,-                                   ;
                                                                                  5$-1$                                     ;
```

```
                                        53            04  D0 0002F 2$:     MOVL    #4, R3                                         ;
                                                      1F  11 00032         BRB     7$                                             ;
                                        53            05  D0 00034 3$:     MOVL    #5, R3                                         ;
                                                      1A  11 00037         BRB     7$                                             ;
                                        53            06  D0 00039 4$:     MOVL    #6, R3                                         ;
                                                      15  11 0003C         BRB     7$                                             ;
                                                  04  A4  DD 0003E 5$:     PUSHL   4(R4)                                          ; 1647
                                           00000000G  8F  DD 00041         PUSHL   #ANLRMS$_BADDATARECPS                          ;
                                 0000G  CF            02  FB 00047         CALLS   #2, ANL$FORMAT_ERROR                           ;
                                                      56  DD 0004C         PUSHL   R6                                             ; 1648
                                        67            01  FB 0004E         CALLS   #1, LIB$SIGNAL                                 ;
                                                      53  D4 00051 6$:     CLRL    R3                                             ; 1643
                                                      53  D6 00053 7$:     INCL    LENGTH                                         ; 1641
                                    14  A4            53  D1 00055         CMPL    LENGTH, 20(R4)                                 ; 1655
                                                      13  1B 00059         BLEQU   8$                                             ;
                                                  04  A4  DD 0005B         PUSHL   4(R4)                                          ; 1656
                                           00000000G  8F  DD 0005E         PUSHL   #ANLRMS$_BADSIDRPTRFIT                         ;
                                 0000G  CF            02  FB 00064         CALLS   #2, ANL$FORMAT_ERROR                           ;
                                                      56  DD 00069         PUSHL   R6                                             ; 1657
                                        67            01  FB 0006B         CALLS   #1, LIB$SIGNAL                                 ;
                                        55        08  AC  E9 0006E 8$:     BLBC    REPORT, 14$                                    ; 1661
                                               0000'  CF  9F 00072         PUSHAB  POINTER_FLAGS_DEF                              ; 1665
                                        7E            62  9A 00076         MOVZBL  (PP), -(SP)                                    ;
                                           00000000G  8F  DD 00079         PUSHL   #ANLRMS$_IDXSIDRPTRFLAGS                       ;
                                                  0C  AC  DD 0007F         PUSHL   INDENT_LEVEL                                   ;
                                 0000G  CF            04  FB 00082         CALLS   #4, ANL$FORMAT_FLAGS                           ;
                         3C             62            04  E0 00087         BBS     #4, (PP), 14$                                  ; 1669
           50            62             02            00  EF 0008B         EXTZV   #0, #2, (PP), R0                               ; 1671
                         02             00            50  CF 00090         CASEL   R0, #0, #2                                     ;
                       0014           000C          0006     00094 9$:     .WORD   10$-9$,-                                       ;
                                                                                   11$-9$,-                                       ;
                                                                                   12$-9$                                         ;
                                        7E        03  A2  3C 0009A 10$:    MOVZWL  3(PP), -(SP)                                   ; 1672
                                                      0B  11 0009E         BRB     13$                                            ;
           7E       03   A2             18            00  EF 000A0 11$:    EXTZV   #0, #24, 3(PP), -(SP)                          ; 1673
                                                      03  11 000A6         BRB     13$                                            ;
                                                  03  A2  DD 000A8 12$:    PUSHL   3(PP)                                          ; 1674
           7E            62             02            00  EF 000AB 13$:    EXTZV   #0, #2, (PP), -(SP)                            ; 1670
                                        6E            02  C0 000B0         ADDL2   #2, (SP)                                       ;
                                        7E        01  A2  3C 000B3         MOVZWL  1(PP), -(SP)                                   ;
                                           00000000G  8F  DD 000B7         PUSHL   #ANLRMS$_IDXSIDRPTRREF                         ;
                                                  0C  AC  DD 000BD         PUSHL   INDENT_LEVEL                                   ;
                                                      7E  D4 000C0         CLRL    -(SP)                                          ;
                                 0000G  CF            06  FB 000C2         CALLS   #6, ANL$FORMAT_LINE                            ;
                                               0000'  CF  9F 000C7 14$:    PUSHAB  POINTER_FLAGS_DEF                              ; 1681
                                        50            62  9A 000CB         MOVZBL  (PP), R0                                       ;
                         7E             50 FFFFFF03  8F  CB 000CE          BICL3   #-253, R0, -(SP)                               ;
                                                  04  A4  DD 000D6         PUSHL   4(R4)                                          ;
                                 0000G  CF            03  FB 000D9         CALLS   #3, ANL$CHECK_FLAGS                            ;
                                    14  A4            53  C2 000DE         SUBL2   LENGTH, 20(R4)                                 ; 1687
                                                      08  13 000E2         BEQL    15$                                            ; 1688
                                    08  A4            53  C0 000E4         ADDL2   LENGTH, 8(R4)                                  ; 1689
                                        50            01  D0 000E8         MOVL    #1, R0                                         ; 1692
                                                          04 000EB         RET                                                    ;
                                                      50  D4 000EC 15$:    CLRL    R0                                             ;
                                                          04 000EE         RET                                                    ; 1694
```

```
; Routine Size:  239 bytes,    Routine Base:  $CODE$ + 093E


; 1209           1695  1
; 1210           1696  0 end eludom



                                                                                 .EXTRN  LIB$SIGNAL

;                                             PSECT SUMMARY
;
;       Name                            Bytes                          Attributes
;
;  $PLIT$                                 186  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $OWN$                                   96  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                                2605  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)




;                                      Library Statistics
;
;                                                   -------- Symbols --------      Pages        Processing
;       File                                        Total    Loaded   Percent      Mapped       Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1                    18619        38         0      1000           00:01.8




; Information:   2
; Warnings:      0
; Errors:        0



;                                             COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RMS3IDX/OBJ=OBJ$:RMS3IDX MSRC$:RMS3IDX/UPDATE=(ENH$:RMS3IDX)

; Size:          2605 code + 282 data bytes
; Run Time:           00:46.8
; Elapsed Time:       02:10.9
; Lines/CPU Min:       2172
; Lexemes/CPU-Min: 20559
; Memory Used:  287 pages
; Compilation Complete
```

0007 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

OB MISC
LIS

RMS2IDX
LIS

RMS3IDX
LIS

RMS
LIS

OB [illegible] R
LIS

0008 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY